

Naval Research Laboratory

Washington, DC 20375-5000



AD-A242 703



DTIC

ELECTE

NOV 21 1991

NRL Report

S C D

A Laser Test Set for the Low-Power Atmospheric Compensation Experiment Satellite

JEANNE A. WELCH, BONNIE LIGHT, AND GARY L. TRUSTY

*Applied Optics Branch
Optical Sciences Division*

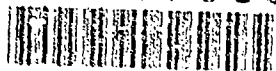
and

THOMAS H. COSDEN

*University Research Foundation
Greenbelt, Maryland*

October 24, 1991

91-16000



Approved for public release; distribution unlimited.

01 1120 000

Naval Research Laboratory

Washington, DC 20375-5000



AD-A242 703



DTIC

ELECTE

NOV 21 1991

NRL Report 9360

S C D

A Laser Test Set for the Low-Power Atmospheric Compensation Experiment Satellite

JEANNE A. WELCH, BONNIE LIGHT, AND GARY L. TRUSTY

*Applied Optics Branch
Optical Sciences Division*

and

THOMAS H. COSDEN

*University Research Foundation
Greenbelt, Maryland*

October 24, 1991

91-16000



Approved for public release; distribution unlimited.

91 1100 000

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE October 24, 1991	3. REPORT TYPE AND DATES COVERED Final Jan 87 — Sept 90		
4. TITLE AND SUBTITLE A Laser Test Set for the Low-Power Atmospheric Compensation Experiment Satellite		5. FUNDING NUMBERS PR - 65-2443-E1 WU - DN157149		
6. AUTHOR(S) J.A. Welch, B. Light, G.L. Trusty, and T.H. Cosden				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5000		8. PERFORMING ORGANIZATION REPORT NUMBER NRL Report 9360		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Strategic Defense Initiative Organization Washington, DC 20301		10. SPONSORING / MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) A laser test set (LATS) was designed for the Low-Power Atmospheric Compensation Experiment (LACE) satellite. LATS provided functional tests for the satellite prior to launch. Laser radiation replicating that to be employed in the satellite experiments irradiated the satellite sensors to qualitatively verify performance. LATS required a rapid, automated goniometer 2-axis pointing system. The algorithm to correct for the tangential distortion that is introduced when a spherical surface is mapped onto a planar target is described in detail.				
14. SUBJECT TERMS Spherical surface mapping Goniometer Tangential geometric distortion			15. NUMBER OF PAGES 52	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR	

CONTENTS

1. INTRODUCTION	1
2. LATS	1
3. POINTING	6
Determination of the Mirror Normal	8
Determination of the Angles to Position the Mirror Normal	8
LATS Calibration	10
4. POWER COMPUTATION	11
5. LATS ALGORITHM	12
Initialization Routine	12
Setup Routine	12
Pointing Routine	13
6. VACUUM TESTS	13
7. CONCLUSIONS	14
8. ACKNOWLEDGMENTS	14
9. REFERENCES	14
APPENDIX A—GPIB Routines for LATS	15
APPENDIX B—Detailed Solutions for Equations (16) and (17)	17
APPENDIX C—LATS Programs	21

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

A LASER TEST SET FOR THE LOW-POWER ATMOSPHERIC COMPENSATION EXPERIMENT SATELLITE

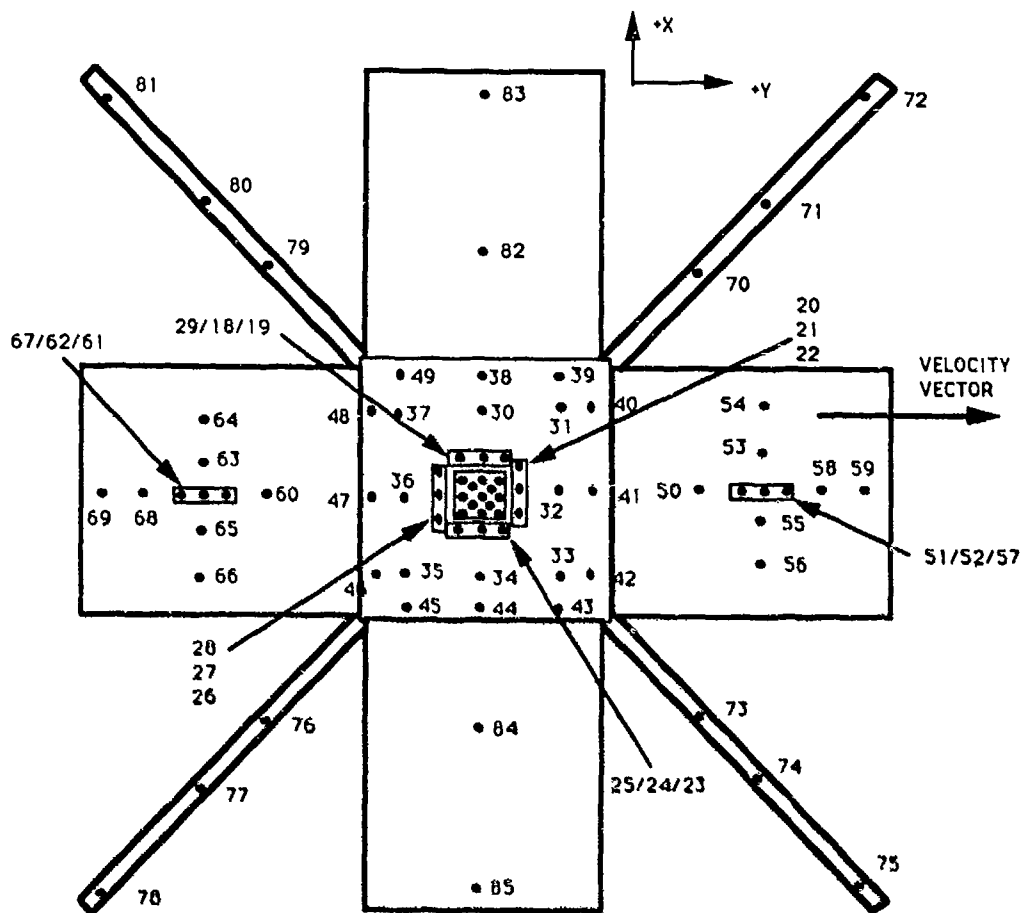
1. INTRODUCTION

The spatial profile of a light beam altered by transmission through atmospheric turbulence can be restored by compensation. Atmospheric compensation techniques rely on knowledge of the altered beam intensity profile. The Low-Power Atmospheric Compensation Experiment (LACE) satellite was developed for proof-of-concept tests for atmospheric compensation of ground-based lasers. The objective of LACE is to measure, in real time, the absolute intensity and spatial distribution of low-energy, atmospherically nondamaging ultraviolet, visible, and infrared radiation transmitted from ground sites to the satellite. An array of 335 sensors is on the face of the satellite that is used to measure the intensity profile of the laser radiation incident on the satellite in orbit. This array comprises 85 sensor pairs for pulsed ultraviolet-visible near-infrared signals, 85 sensors for acousto optically modulated visible near-infrared signals, and 40 sensor pairs for mechanically chopped infrared signals. Figure 1 schematically shows the sensor array.

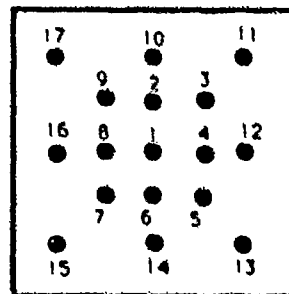
Prior to launch, two functional tests were required for sensor and electronics performance verification: sensor integration in a model spacecraft (integration test) and operational evaluation in the completed spacecraft under vacuum and thermal cycling (vacuum test). Because of the large number of sensors and the short satellite operating times imposed by cooling requirements, it was imperative that irradiation of each sensor occur rapidly and automatically. To conduct these tests, a laser test set (LATS) was developed. The principal function of the test system was to irradiate the sensor array subsystem (SAS) with laser radiation replicating that to be employed in the LACE experiments. This provided information to the LACE ground computer on laser power levels at the individual sensors to verify wavelength response and dynamic range of the SAS. LATS was not designed to quantitatively calibrate sensor parameters but to qualitatively verify sensor performance. Both individual sensor tests (small cross-section beams scanned over the face of the satellite) and multisensor flood tests of the entire satellite face were required for validation. Repeated irradiation of each individual sensor required a precise and reproducible pointing scheme. Flood tests of the entire sensor array required a removable lens/mirror combination to produce an expanded beam. The LATS was used in the integration test, and then it was modified for use in the vacuum tests.

2. LATS

Figure 2 is a schematic drawing of the LATS configuration. The elements of LATS included a set of lasers, a transfer mirror, a goniometer-mounted steering mirror and stepping-motor controller, a power meter, a computer, and a network of bidirectional data buses. The laser beam was expanded and collimated to a 2-in. diameter to provide a uniform beam profile. To provide an intensity reference, a beamsplitter reflected a portion of the light onto a power meter. The computer recorded the reference power by using an analog-to-digital (A/D) converter. The transmitted portion of the laser beam was reflected from a transfer mirror (TM) located in front of the plane of the sensor panel onto the goniometer-mounted steering mirror (SM). The computer then aimed the steering mirror to direct the laser beam onto the specified target sensor.



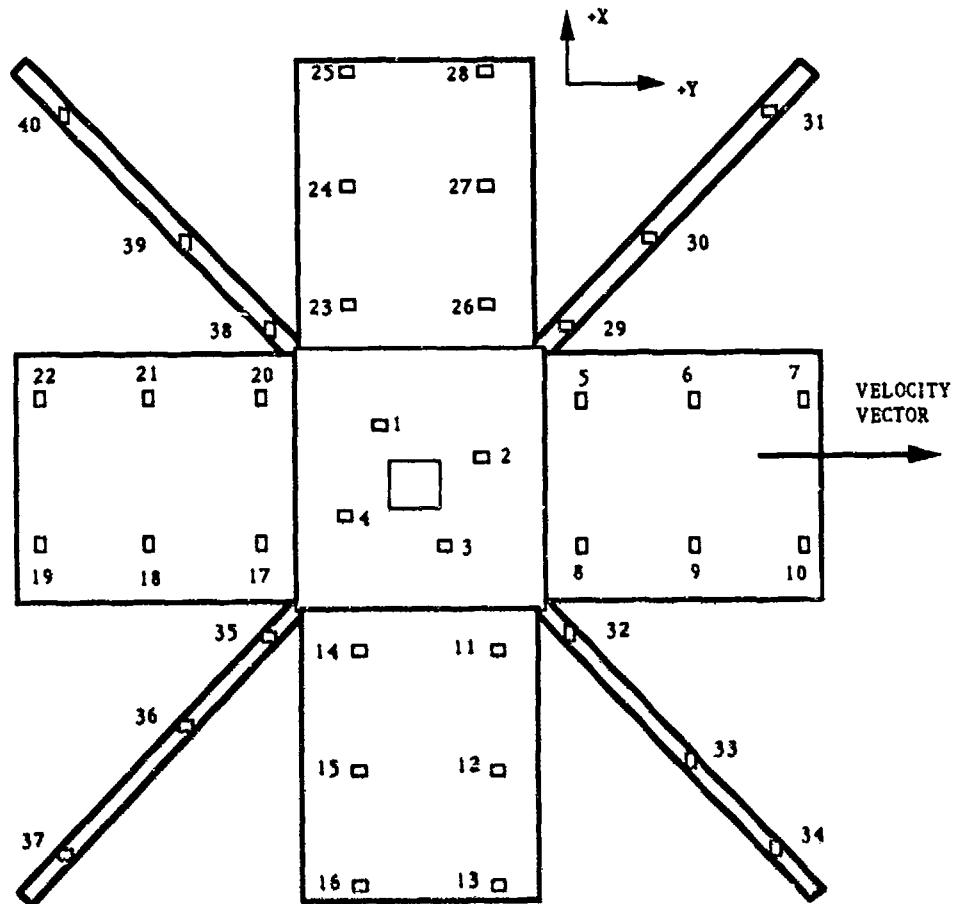
VIEWED FROM EARTH



CENTRAL ARRAY

(a) Acousto-optically modulated and pulsed sensor positions

Fig. 1 - Schematic drawing of LACE sensor array



VIEWED FROM EARTH

(b) Mechanically-modulated infrared sensor positions

Fig. 1 (Continued) — Schematic drawing of LACE sensor array

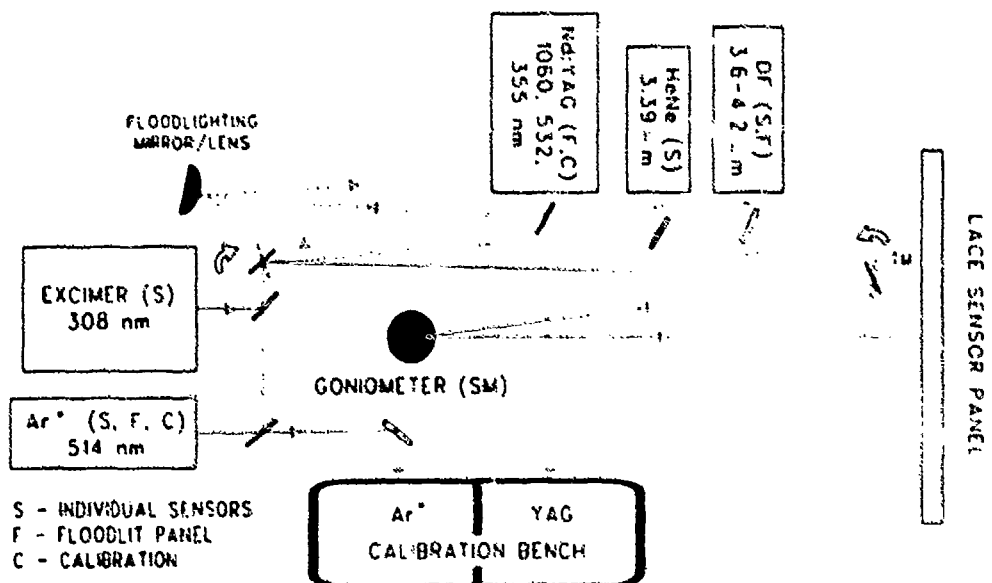


Fig. 2 — Schematic drawing of LATS configuration used for the integration tests

Generation of the required radiation was accomplished with five lasers: 25 Hz excimer and 10 Hz Nd:YAG lasers were the pulsed-radiation sources; 25 kHz acousto-optically modulated argon ion, 1440 Hz mechanically chopped helium neon (HeNe), and 1440 Hz mechanically chopped deuterium fluoride (DF) lasers were the continuous-wave sources. The LACE test plan [1] describes the radiation specifications and Tables 1 and 2 summarize them.

Table 1 — Laser Characteristics

Laser	Wavelength (μm)	Beam Diameter (mm)	Maximum Power	PRF (Hz)	Pulse Length (ns)
Excimer	0.308	22	90 mJ	25	19
Ar ⁺	0.5145	3	250 mW	25k	—
Nd:YAG	0.355	8	250 mJ	10	10
Nd:YAG	1.064	8	1 J	10	10
DF	3.6-4.2	5	1 W	1440	
HeNe	3.39	5	5 mW	1440	

Table 2 — LATS Integration Requirements

Laser	Wavelength (μm)	Power Density (W/cm^2)	
		Maximum	Minimum
Excimer	0.308	10^4	10^2
DF	3.6-4.2	10^{-4}	5×10^{-6}
Ar ⁺	0.5145	10^{-4}	10^{-6}
Nd:YAG	0.355	10^4	10^2
Nd:YAG	1.064	10^3	10^1

The goniometer was a Klinger BG 120 YZ rotary cradle mounted on an RT 120 XZ rotary stage with a Klinge CC-1 programmable indexer and a step resolution of 0.01° . Both the stage and cradle had an incremental encoder for precise closed-loop positioning and speed control. These two rotary stages are combined to produce independently controlled motion in azimuth and elevation. The goniometer mirror was in a two-axis gimble mount positioned with the mirror center at the intersection of the goniometric axes. Two independent stepper-motor channels controlled the motion of the two orthogonal axes and allowed for manual as well as programmed positioning. The stepper-motor controller is able to establish a home reference position, make absolute movements relative to that position, or move incrementally relative to a given position.

For LATS, timing control was provided by an electronic circuit rather than the specified LACE trigger laser. Sync-out pulses from the pulsed lasers and motor-driven choppers provided reference signals to trigger sensor panel data collection. The timing circuit could function at any of the necessary detection frequencies. No timing circuit was necessary for the acousto-optically chopped laser because of the short modulation times; a data pulse was always present in the detection window.

The LATS computer was an IBM PC-AT interfaced with two IEEE-488 buses. IBM general purpose interface bus (GPIB) adapters were installed in the PC for each required interface. Through the GPIB, the computer communicated with the goniometer controller, the A/D converter, and the LACE microvax sensor array subsystem test unit (SASTU) as Fig. 3 schematically shows. Two independent communication buses were used in the PC to avoid conflicts in the working environments and to reduce processing time. The first bus served as a data link between the LATS PC and the SASTU. The second bus linked the PC to the stepping-motor controller and the Hewlett-Packard 59313A A/D converter. The SASTU served as system controller-in-charge (CIC) on the first bus. The IBM PC-AT LATS computer was controller on the second bus and operated the two independent axes of the goniometer and the A/D converter as three independent devices. Table 3 summarizes the contents of the configuration files (IBCONF) for the devices used with the GPIB bus. Reference 2 provides a description of GPIB control. Appendix A lists the required subroutine sequence for goniometer movement, data acquisition, and data transfer.

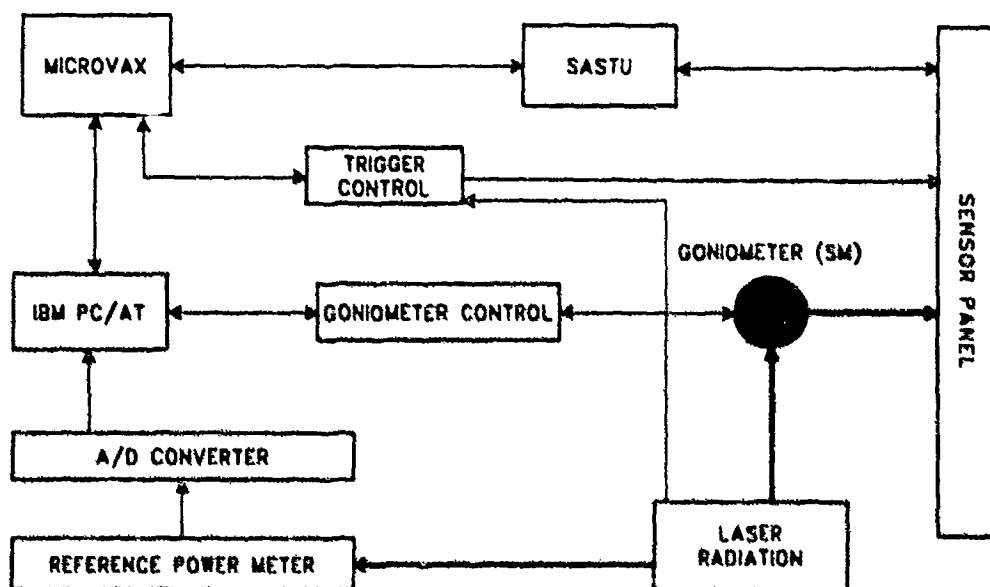


Fig. 3 --Schematic drawing of LATS system. The laser beam is shown in bold.

Table 3 -- GPIB Configuration Parameters for LATS

Device	Address	Time Out (s)	EOS* Byte	CIC	EOI†	Access
GPIB0	00H	3	01H	YES	YES	
GPIB1	01H	none	0AH	NO	YES	
XAXIS	06H	3	01H		YES	GPIB0
YAXIS	07H	3	01H		YES	GPIB0
ADC	08H	10	01H		YES	GPIB0
MVAX		3	0AH		YES	GPIB1

*End of string

†Send EOI (end of data message) with EOS

Programmed control consisted of an ASCII-coded set of high-level instruction commands written to the goniometer controller from the LATS computer over the interface bus. These ASCII strings controlled goniometer device velocity, acceleration, displacement, directionality, scanning interval, and read/write information to the devices. The CC-1 stepper-motor controller manual gives a full description of the programmed and manual operation of the stepper-motor controller.

3. POINTING

Initially, it was thought that accurate pointing could be achieved by evaluating the simple planar trigonometric relations governing a point source incident on a flat target. The computation of the tangent of the enclosed angle, the ratio of the vertical and horizontal distances of the sensor from the origin, and the distance from the steering mirror to the target panel yielded azimuth and elevation angles corresponding to the pointing vector for a given sensor location. A scaled-down laboratory demonstration revealed the large inaccuracies of this simple approach.

The task of pointing a beam steered by an azimuth-elevation device onto a planar surface at a close distance is to map a spherical surface onto an intersecting plane. Since the sensors on the test panel were arranged in a flat plane indexed by Cartesian coordinates and the motion of the incident light swept out spherical contours, a tangential geometric distortion was introduced, and it was necessary for an appropriate mapping to be derived.

Figure 4 shows the laboratory coordinate system used to characterize the mapping of a spherical surface onto a planar surface. The origin is at the center of the steering mirror, coincident with the goniometer origin; angles are measured in the right-handed sense and the z-direction toward the target is negative.

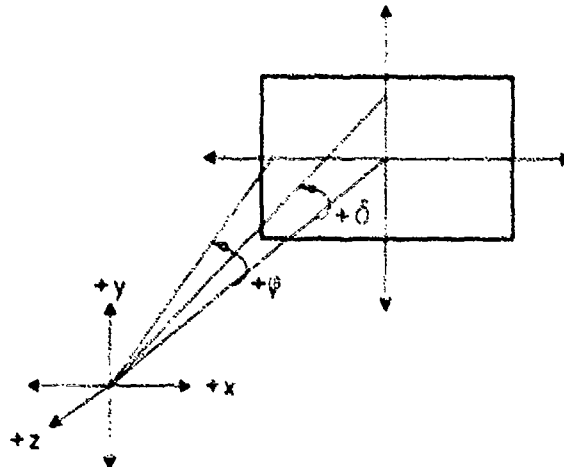


Fig. 4 — Laboratory coordinate system used for LATS

For notation purposes, the generalized vector \bar{v} is defined in terms of the unit vector \bar{v} as

$$\bar{v} = v_x \hat{x} + v_y \hat{y} + v_z \hat{z} = (v_x, v_y, v_z)^T \quad (1)$$

$$\bar{v} = v\hat{v} = v(\theta_x, \theta_y, \theta_z)^T, \quad (2)$$

where $(v_x, v_y, v_z)^T$ is the notation for the transpose of the column vector for matrix manipulations. The magnitude of the vector v is given by

$$v = (v_x^2 + v_y^2 + v_z^2)^{1/2}. \quad (3)$$

In general, as Fig. 5 shows, a reflection of a mirror is described with three vectors. \vec{s} is the vector from the steering mirror to the source, \vec{r} is the vector from the steering mirror to the target, and, \vec{n} is the mirror normal vector that is the coplanar bisector of the angle formed by \vec{s} and \vec{r} shown in Fig. 5.

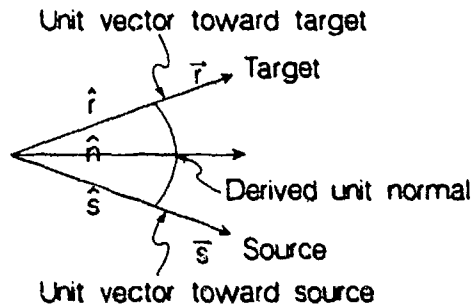


Fig. 5 — Schematic representation of the vectors for LATS pointing algorithm

Two steps are required to determine the transformation to map from the goniometric system to a planar surface. The first step is to find where the mirror normal \hat{n} should be located to reflect light off the goniometer mirror from the source at \hat{s} to a target at \hat{r} , and the second step is to find the angles (ϕ , δ) that will position the goniometer steering mirror normal at this \hat{n} .

The accuracy of the pointing algorithms depends on the accuracy with which the experiment components are located and oriented in the laboratory system. For the goniometer, this is implicit and fixed. By design its coordinates define the laboratory system, and other LATS components must be calibrated with respect to it.

Ideally, the target face is perfectly flat, is perpendicular to the laboratory negative z -axis, and has x - and y -axes parallel to the laboratory axes defined by the goniometer. The target face flatness is least amenable to calibration. In the LATS, the target face is the sensor panel of a precisely machined satellite that is large and massive but not under unreasonable strain. The width of the beam relative to the size of the irradiated sensor area made target flatness a negligible correction.

Perpendicularity of the target face and the negative laboratory z -axis is achieved to adequate accuracy through mechanical placement of the two systems. This also ensures collinearity of the target and goniometer origins. However, it is also a simple matter to redefine the target face coordinate system to lie on the laboratory z -axis. Then the new target sensor positions are computed from the new target origin offset.

Finally, the target face x - and y -axes must be parallel to the laboratory axes and not rotated by angle α about the z axis. Two approaches are used to correct this error: position the target and the goniometer so that the axes are parallel, or measure and correct for a relative displacement by rotating the desired target face (x_T, y_T) coordinates through $-\alpha$ into the laboratory frame coordinates (x_L, y_L) and pointing the beam at (x_L, y_L) instead of (x_T, y_T) for each sensor:

$$\begin{bmatrix} x_L \\ y_L \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} x_T \\ y_T \end{bmatrix} \quad (4)$$

Either approach requires calibrating the source steering mirror as discussed in the LATS calibration section below.

To measure the rotational misalignment, direct the goniometer to a convenient (x, y) position in the laboratory frame. $(0, 0)$ is suggested to simplify the expressions. Rearranging the expression in Eq. (4) yields the following expressions for α . At $x = 0$,

$$\sin \alpha = \frac{x_T}{y}, \quad (5)$$

where y is the measured target frame coordinate, and at $y = 0$,

$$\sin \alpha = -\frac{y_T}{x}. \quad (6)$$

Similarly, x is measured in the target frame coordinate system. The beam is pointed toward large excursions along the x - and y -axes, and deviations x_T and y_T from the targets are measured and used to compute α . If a nonzero α is determined, target frame coordinates are transformed to the laboratory-parallel target frame by using Eq. (4).

Determination of the Mirror Normal

The desired normal orientation of the mirror normal vector, in terms of components along the laboratory axes, is given as the coplanar bisector of \vec{r} and \vec{s} ,

$$\vec{n} = \frac{\vec{r} + \vec{s}}{\sqrt{2n}} = \left[\frac{1}{2n} \right] (\vec{r}_x + \vec{s}_x, \vec{r}_y + \vec{s}_y, \vec{r}_z + \vec{s}_z)^T = (n_x, n_y, n_z)^T, \quad (7)$$

where

$$n = (\vec{n} \cdot \vec{n})^{\frac{1}{2}} = (1/\sqrt{2})(1 + \vec{r}_x \vec{s}_x + \vec{r}_y \vec{s}_y + \vec{r}_z \vec{s}_z)^{\frac{1}{2}}. \quad (8)$$

Because the source beam is reflected one or more times before impinging on the steering mirror at the goniometer origin the light source \vec{s} is located in the laboratory frame at the center of the TM as Fig. 2 indicates. The vector \vec{r} is defined similarly as the sensor target position in the laboratory frame, with the origin of the vector \vec{r} at the center of the steering mirror and the target at the center of the sensor. By measuring the source and target vectors, the terms s , \vec{s} , r , and \vec{r} are easily computed from Eqs. (2) and (3), and the mirror normal is then determined from Eqs. (7) and (8).

This approach accounts for arbitrary misalignments in almost all angles and positions with the exception of two: the steering mirror must be at the origin of the goniometer so that the vertex of the reflection is at the center of rotation, and the transfer mirror is presumed to reflect the light beam directly onto the origin. Arbitrary misalignments are accommodated as long as the angles required to reflect the laser beam from the sensor array origin to the source can be measured accurately.

Determination of the Angles to Position the Mirror Normal

Finally, the angles (ϕ, δ) that will position \vec{n} for the chosen target must be determined. These angles will be dependent on where the mirror normal is when $(\phi, \delta) = (0, 0)$. This is defined as the initial orientation mirror normal, \vec{n}_0 . The goniometer angles (ϕ_0, δ_0) are the angles that reflect the laser beam from the origin directly back to the source \vec{s} on the transfer mirror. Again, \vec{n}_0 and (ϕ_0, δ_0) are measured in the laboratory frame.

The standard matrix representation of a rotation of a unit vector \hat{v} through δ about x is

$$\hat{R}(\delta)\hat{v} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\delta & -\sin\delta \\ 0 & \sin\delta & \cos\delta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \cos\delta - v_z \sin\delta \\ v_y \sin\delta + v_z \cos\delta \end{bmatrix}. \quad (9)$$

Similarly, for a rotation through ϕ about y ,

$$\hat{R}(\phi)\hat{v} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_x \cos\phi + v_z \sin\phi \\ v_y \\ -v_x \sin\phi + v_z \cos\phi \end{bmatrix}. \quad (10)$$

Note that the elevation rotation leaves x unchanged, and the azimuthal rotation leaves y unchanged, as these are the corresponding orthogonal rotation axes. The combined rotation is given by the product matrix,

$$\hat{R}(\phi, \delta)\hat{v} = \hat{R}(\phi)\hat{R}(\delta)\hat{v} = \begin{bmatrix} \cos\phi & \sin\phi \sin\delta & \sin\phi \cos\delta \\ 0 & \cos\delta & -\sin\delta \\ -\sin\phi & \cos\phi \sin\delta & \cos\phi \cos\delta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}. \quad (11)$$

$\hat{R}^{-1}(\phi, \delta) = \hat{R}^T(\phi, \delta)$ for this orthogonal matrix, thus simplifying the matrix transformations required to compute the mirror normal,

$$\hat{R}(\phi, \delta) \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = (\hat{R}_x, \hat{R}_y, \hat{R}_z) \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_x \cos\phi + v_y \sin\phi \sin\delta + v_z \sin\phi \cos\delta \\ v_y \cos\delta - v_z \sin\delta \\ -v_x \sin\phi + v_y \cos\phi \sin\delta + v_z \cos\phi \cos\delta \end{bmatrix}. \quad (12)$$

To determine what angles are required to direct the source beam onto the target, the rotation that carries the initial mirror normal $\hat{A}_0 = (A_{0x}, A_{0y}, A_{0z})^T$ into the desired mirror normal $\hat{A} = (A_x, A_y, A_z)^T$ is determined from Eq. (12):

$$A_x = A_{0x} \cos\phi + A_{0y} \sin\phi \sin\delta + A_{0z} \sin\phi \cos\delta \quad (13)$$

$$A_y = A_{0y} \cos\delta - A_{0z} \sin\delta \quad (14)$$

$$A_z = -A_{0x} \sin\phi + A_{0y} \cos\phi \sin\delta + A_{0z} \cos\phi \cos\delta. \quad (15)$$

The trigonometric functions in ϕ and δ given in Eqs. (13) and (15) complicate rearrangement into closed expressions for the angles themselves; some trigonometric substitutions and the Pythagorean relation are required for solution; the details are described in Appendix B.

$$\delta = \cos^{-1} \left[\frac{\hat{n}_y}{(1 - \hat{n}_{0x}^2)^{1/2}} \right] - \cos^{-1} \left[\frac{\hat{n}_{0y}}{(1 - \hat{n}_{0x}^2)^{1/2}} \right] \quad (16)$$

$$\phi = \cos^{-1} \left[\frac{\hat{n}_z}{(1 - \hat{n}_y^2)^{1/2}} \right] - \cos^{-1} \left[\frac{(\hat{n}_{0y} \sin \delta + \hat{n}_{0z} \cos \delta)}{(1 - \hat{n}_y^2)^{1/2}} \right]. \quad (17)$$

When the goniometer is positioned at (ϕ, δ) , light from the transfer mirror, positioned at (s_x, s_y, s_z) and incident on a mirror at the goniometer origin at orientation (n_{0x}, n_{0y}, n_{0z}) when $(\phi, \delta) = (0, 0)$, is reflected to the target position (r_x, r_y, r_z) .

The mirror normal vector required to direct the source onto the target is easily computed from the values of the source and target normal vectors

$$\hat{n}_i = \frac{\hat{r}_i + \hat{s}_i}{2} \quad i = x, y, z \quad (18)$$

and normalizing the resultant vector.

LATS Calibration

It is important to precisely determine steering mirror position and to accommodate imperfect goniometer mirror mounting. The following procedure does not depend on the presumption of target flatness nor on the determination and elimination of the rotational alignment error α that are both zero at the origin. It is an iterative computation dependent on the two sets of goniometer angles (ϕ, δ) that reflect the source beam onto the target origin and the steering mirror in the laboratory frame.

The steering mirror is placed in the $-z$ hemisphere, as close angularly to the target origin as possible without eclipsing a sensor. Since the goniometer mirror is to reflect the beam at twice the normal angle of incidence, there is an angular precision bias between targets on the source mirror side of the target origin and targets on the opposite side of the target origin. Angular inaccuracy and imprecision are doubled by the reflection and are proportional to the source-goniometer/target-angle magnitude. Therefore, all angles are minimized by keeping the steering mirror angularly close to the target origin. If the steering mirror cannot be located angularly coincident with the target origin, the alternatives are to locate it in the xz -plane eliminating y bias, or in the yz -plane eliminating x bias. Because of the noncritical accuracy requirements of the present experiment, the steering mirror was placed approximately 8° from the target origin.

Two simple measurements on the LATS are sufficient to permit an iterative computation of the mirror normal that reflects a source beam onto the target origin at $(0, 0, -z)$. These two measurements are the goniometer angles required to reflect the source beam onto the target origin, and the angles required to reflect the source beam back onto itself. Despite inaccuracies in initial estimates of both the source position and the at-rest goniometer mirror normal \hat{n}_0 , the prescribed measurements contain sufficient information about the angle to converge numerically to a self-consistent and correct value of both position vectors simultaneously.

It is convenient but not important that (ϕ, δ) be nearly zero. The goniometer used in the LATS supported a logical definition of $(\phi, \delta) = (0, 0)$ that was used. This constitutes the initial orientation of the goniometer mirror normal. The estimated position of the source (x, y, z) is measured in the laboratory frame. The vector is normalized and the result designated \hat{s}_0 where the subscript denotes an initial approximation to the true source normal. It is precisely because of the difficulty of measuring both of these critical parameters accurately that the method described is used.

The first iterative correction to the initial mirror normal vector pointing at the sensor array origin, $r = (0, 0, -1)$, is given as

$$\hat{R}_0 \hat{A}_0 = (2 - 2\hat{s}_{0z}) (\hat{s}_{0x} - 0, \hat{s}_{0y} - 0, \hat{s}_{0z} - 1) = (2 - 2\hat{s}_{0z}) (\hat{s}_{0x}, \hat{s}_{0y}, \hat{s}_{0z} - 1) \quad (19)$$

from Eq. (7), with \hat{R}_0 the rotation given by Eq. (12) that reflects the source beam onto the origin. This new \hat{A}_0 is used in Eqs. (13) and (15) to determine a new \hat{s}_0 from the measured angles (ϕ_0, δ_0) required to reflect the initial mirror normal onto the source. These values, in turn, are used to iteratively determine a new initial mirror normal in Eq. (19) until the solution converges.

Because of the angle-doubling effect of beam reflection, an error in the mirror normal causes twice the error in the reflected beam. Conversely, an error in the reflected beam's measurement corresponds to half the error in the mirror normal. Beginning the iterative procedure with the right-hand-side of Eq. (19) supports convergence because errors in the position of \hat{s} are halved before correcting with Eqs. (13) and (15) on the next iteration. Convergence is guaranteed if the experiment is designed with the goniometer at-rest mirror normal \hat{A}_0 , not the source position \hat{s} , which is angularly closer to the laboratory target origin. Then the rotation required to reflect the mirror normal onto the target origin is smaller than that required to reflect the mirror normal onto the source. For the nominal position $\hat{A}_0 = \hat{r}_0$, this error is halved, which roughly cancels the angle-doubling error introduced in Eqs. (13) and (15). Application of Eq. (19) halves errors while application of Eqs. (13) and (15) preserves them, and the procedure converges.

Implementation of this goniometric mapping in the laboratory demonstrated a large improvement over the initial approach. However, the accuracy of the mapping is a function of the accuracy of the input parameters, including presumptions of perpendicularity of the target plane and the laboratory z-axis. For the supplied, not measured, sensor locations used here, each position was indexed by its xy distance from the satellite origin and specified to accuracies of 1.0 cm for the infrared sensors and 0.5 cm for the ultraviolet and visible sensors; errors larger than those specified are thought to occur for several sensor locations in the model spacecraft used for the integration tests. These unexpectedly large inaccuracies in sensor placement from the indexed Cartesian coordinates required the mapping to be augmented with manual position fine tuning to establish the final goniometric pointing vector.

4. POWER COMPUTATION

Incident power computations were made by the LATS system and sent to the SASTU for comparison with the powers collected at the sensors. The ratio of the beam power reflected by the beamsplitter to that transmitted through the optical path to the target R_b was known with accuracies of 10% for each wavelength used. The computation of incident power required knowledge of laser and sensor-dependent parameters. These parameters included any neutral density filtering in the laser beam (ND), the pulse repetition frequency (PRF) of the laser, the pulse length (PL) of the laser, and the ratio of the areas of

the sensor and the laser beam. Equations (20) and (21) give the incident power for the chopped and pulsed sensors, respectively:

$$P_{chopped} = P_{in} \cdot 10^{-ND} \cdot R_b \cdot \frac{Area_{sensor}}{Area_{beam}}, \quad (20)$$

$$P_{pulsed} = P_{in} \cdot 10^{-ND} \cdot PRF^{-1} \cdot PL^{-1} \cdot \frac{Area_{sensor}}{Area_{beam}}. \quad (21)$$

P_{in} is the calibrated power reading from the analog to digital converter. To optimize the placement of the laser beam at each sensor following goniometric and manual pointing, the beam was scanned over a $0.25^\circ \times 0.25^\circ$ grid in 5×5 steps centered on the sensor. Average power levels are measured, and the position yielding the highest power is retained for the final vector.

5. LATS ALGORITHM

Three computer programs were used for the LATS tests. An initialization routine created and formatted the five sensor-dependent position information bases for the 335 satellite sensors into five sensor-dependent location files (pulsed high power, pulsed low power, modulated, infrared high power, and infrared low power). The infrared and ultraviolet sensors are distinct detector pairs that cover well-defined power ranges. A setup routine calculated the goniometer pointing vectors, allowed manual vector correction, and wrote the final vector database. Finally, a pointing routine interfaced with the SASTU passed sensor identification numbers, pointed the laser beam at the specified sensor, and passed power data back to the SASTU. All programs were written in RM Professional Fortran [3], and the setup and pointing routines are included in Appendix C.

Initialization Routine

The following describes the procedures in the initialization routine.

- A. Prompt user for input laser identification number to distinguish among the five sensor-dependent location files.
- B. Input Cartesian coordinates of the sensor with respect to the satellite origin.
- C. Write the indexed coordinates to the sensor-dependent location file.

Setup Routine

The following describes the procedures in the goniometer setup routine.

- A. Prompt user for the sensor identification number and laser type.
- B. Look up indexed position information from sensor location file.
- C. Compute and calibrate pointing angles.
 1. Compute rest mirror normal \hat{n}_0 from steering mirror and target vectors.
 2. Compute desired normal orientation \hat{n} from Eq. (7) knowing the target vector and the source vector. The initial mirror normal is computed by using the solution \hat{n}_0 to recompute \hat{n} that is then solved again for the mirror normal until the solutions converge.
 3. Compute elevation and azimuth angles for sensor location from Eqs (16) and (17).

- D. Direct the laser beam onto the specified sensor.
 - 1. Convert angular information into step interval and form ASCII string commands.*
 - 2. Write the command string to the goniometer.
- E. Prompt user to manually correct laser beam placement on the sensor.
- F. Scan across sensor face to locate laser beam placement for maximum sensor irradiance (optional routine).
- G. Record position of maximum irradiance on position vector database.

Pointing Routine

The following describes the procedures followed in the goniometer pointing routine.

- A. Read sensor identification from SASTU.
- B. Obtain sensor azimuth and elevation angles from vector information database.
- C. Direct laser beam on the specified sensor.
- D. Record incident laser beam power.
- E. Loop to A.

Upon completion of the automated individual sensor tests, a flood test was performed at each wavelength. By reflecting the laser beam from the transfer mirror to the convex mirror/concave lens assembly illustrated in Fig. 2, the beam was expanded to irradiate all sensors on the satellite simultaneously for the multisensor flood tests. These tests, used to determine whether there is interference between sensors, do not require precision pointing or computer control.

6. VACUUM TESTS

Figure 6 shows several modifications that were made on the LATS for the vacuum chamber tests. A doubled Nd:YAG laser, an argon-ion laser, and an infrared quartz-halogen lamp were the three sources used for the vacuum tests. Because of mechanical limitations, the infrared lamp was used only for flood testing of the sensor array, and the two visible lasers were used for both multisensor flood testing and individual sensor irradiation. Interchangeable lenses placed in the entrance port to the vacuum chamber provided the collimated or diverging laser beam for the two types of tests, individual and flood.

Long periods of time were required to attain the vacuum in the large satellite chamber. To avoid having to open the chamber between thermal cycles or between individual/flood tests modes to reestablish the vacuum, a remotely controlled method of switching between multi-sensor flood testing and individual sensor testing was devised. A convex mirror mounted to the backside of the goniometer gimble mount allowed one hemisphere of the goniometer to reflect collimated laser light and the other hemisphere to reflect rapidly diverging light. To switch between the two types of tests, the goniometer carousel had

*To write a string to the goniometer controller the length of the string must be specified. It is necessary for this program, as written in RM Professional Fortran, to branch into individual subroutines to accommodate the string size of the goniometer command.

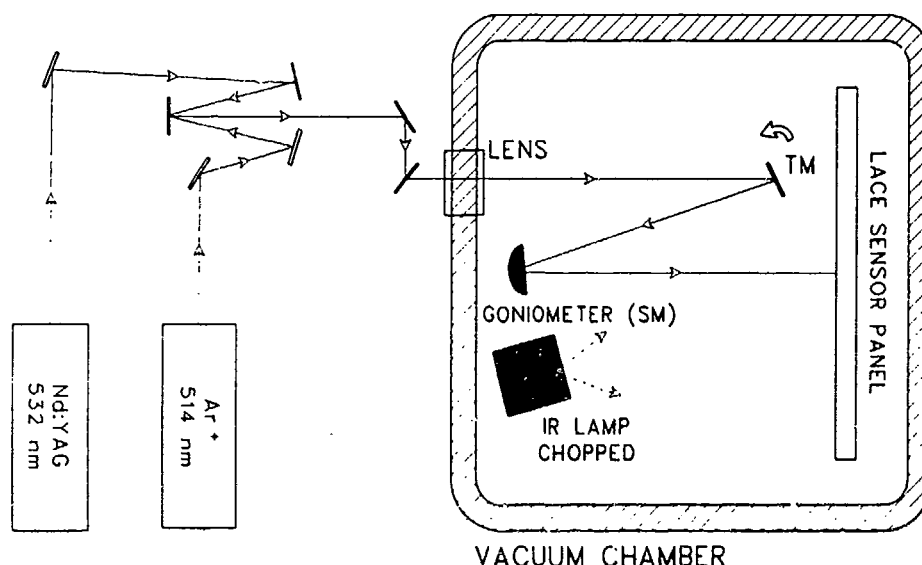


Fig. 6 — Schematic drawing of LATS configuration used for the vacuum tests

only to be rotated by 180° . The infrared lamp was in a pressurized canister and positioned in the vacuum chamber to provide complete irradiation of the satellite face. Except for the modifications just described, validation tests for the scanned individual sensors and multisensor tests equivalent to those outside the chamber were performed on the spacecraft in the vacuum chamber.

7. CONCLUSIONS

The LATS system provided precise, repeatable, and automated goniometric vectoring for preflight system integration and operational evaluation of the LACE spacecraft. The pointing vector algorithm developed for the goniometer provided placement of a laser beam at any sensor location on the satellite, based only on the indexed Cartesian coordinates of the sensor and the distance of the goniometer steering mirror and transfer mirror from the satellite sensor plane origin. Large errors in sensor location from the indexed coordinates were corrected with manual vector positioning. Once pointing vectors were established for all satellite sensor locations, automated laser testing was performed.

8. ACKNOWLEDGMENT

Many discussions of the pointing problem with Dr. Scott Wallace are gratefully acknowledged.

9. REFERENCES

1. "Low Power Atmospheric Compensation Experiment Requirements Document," SSD-D-IL001, Revision 5, Jan. 1987.
2. IBM Personal Computer General Purpose Interface Bus Adapter Programming Support. Boca Raton, FL: IBM.
3. RM/FORTRAN Version 2.4. Rolling Hills Estates, CA: Ryan-McFarland.

Appendix A

GPIB ROUTINES FOR LATS

GPIB Routine for LATS Goniometer Motion

- C move x-axis n steps (n = two digit integer, e.g. 10)
integer*2 xaxis
character*5 X\$
- C assign unit descriptor for x-axis
xaxis = ibfind ('xaxis')
- C clear the device
call ibclr (xaxis)
- C file "RATE" contains goniometer velocity and acceleration values
- C write file "RATE" to the x-axis
call ibwrtf (xaxis, rate)
- C clear the device
call ibclr (xaxis)
- C construct string containing N number of steps
X\$ = 'N'/' '/'10'/'char(13)
- C write X\$ to x-axis
call ibwrt (xaxis, X\$, 5)

GPIB Routine to Read the Analog to Digital Converter

- C read converted measurement from HP converter
integer*2 adc
character*2 buffera
- C assign unit descriptor for analog to digital converter
adc = ibfind ('adc')
- C command for converter to write data is 'H8AJ'
call ibwrt (adc, 'H8AJ', 4)
- C data is written as binary-coded decimal in ASCII
call ibrd(adc, buffera, 2)

GPIB Routine to Write and Read to/from the SASTU

- C assign unit descriptor for adapter linked to SASTU
integer*2 gp
character*10 bufferb, bufferc
gp = ibfind ('GPIB2')
- C write data to GPIB adapter
call ibwrt (gp, bufferb, 10)
- C read data from the GPIB adapter
call ibrd (gp, bufferc, 10)

Appendix B

DETAILED SOLUTION FOR EQUATIONS (16) AND (17)

To solve for ϕ and δ , three equations are used in two unknowns. These trigonometric representations must be arranged into closed expressions for the angles. By using trigonometry and the generalized Pythagorean relation

$$q_x^2 + q_y^2 + q_z^2 = 1, \quad (\text{B1})$$

these expressions can be found.

From Eq. (14),

$$A_y = A_{0y} \cos \delta - A_{0x} \sin \delta. \quad (\text{B2})$$

By using the trigonometric construction shown in Fig. B1 and making the following substitutions, $a = A_{0y}$, $b = A_{0x}$ yields

$$A_y = a \cos \delta - b \sin \delta \quad (\text{B3})$$

$$= c (\cos \alpha \cos \delta - \sin \alpha \sin \delta) \quad (\text{B4})$$

$$= c \cdot \cos(\alpha + \delta) \quad (\text{B5})$$

by trigonometric identity;

$$A_y = (a^2 + b^2)^{1/2} \cos(\alpha + \delta) \quad (\text{B6})$$

$$A_y = (A_{0y}^2 + A_{0x}^2)^{1/2} \cos(\alpha + \delta). \quad (\text{B7})$$

$$c = (a^2 + b^2)^{1/2}$$

$$\cos \alpha = a/c$$

$$\sin \alpha = b/c$$

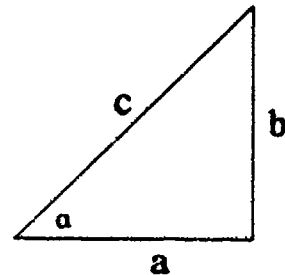


Fig. B1 — Trigonometric construction needed to simplify the expressions used to solve for Eqs. (16) and (17)

By using the identity in Eq. (B1) and rearranging:

$$\cos(\alpha + \delta) = \frac{n_y}{(1 - n_{ox}^2)^{1/2}} \quad (B8)$$

$$\delta = \cos^{-1} \left[\frac{n_y}{(1 - n_{ox}^2)^{1/2}} \right] - \alpha \quad (B9)$$

$$\delta = \cos^{-1} \left[\frac{n_y}{(1 - n_{ox}^2)^{1/2}} \right] - \cos^{-1} \left[\frac{n_{oy}}{(1 - n_{ox}^2)^{1/2}} \right]. \quad (B10)$$

This is a closed-form expression for δ .

From Eq. (15),

$$n_z = -n_{ox} \sin \phi + n_{oy} \cos \phi \sin \delta + n_{oz} \cos \phi \cos \delta \quad (B11)$$

$$= \cos \phi (n_{oy} \sin \delta + n_{oz} \cos \delta) - \sin \phi (n_{ox}), \quad (B12)$$

which is of the same form as Eq. (B2). Letting $a = (n_{oy} \sin \delta + n_{oz} \cos \delta)$ and $b = n_{ox}$, rearranging the expression yields

$$n_z = \left[(n_{oy} \sin \delta + n_{oz} \cos \delta)^2 + n_{ox}^2 \right]^{1/2} \cos(\phi + \beta). \quad (B13)$$

Simplification is possible by expanding the square-root term as follows:

$$(n_{oy} \sin \delta + n_{oz} \cos \delta)^2 + n_{ox}^2 = n_{oy}^2 \sin^2 \delta + n_{oz}^2 \cos^2 \delta + 2n_{oy}n_{oz} \sin \delta \cos \delta + n_{ox}^2 \quad (B14)$$

$$= n_{oy}^2(1 - \cos^2 \delta) + n_{oz}^2(1 - \sin^2 \delta) + n_{ox}^2 + 2n_{oy}n_{oz} \sin \delta \cos \delta \quad (B15)$$

$$= (n_{ox}^2 + n_{oy}^2 + n_{oz}^2) + 2n_{oy}n_{oz} \sin \delta \cos \delta - n_{oy}^2 \cos^2 \delta - n_{oz}^2 \sin^2 \delta \quad (B16)$$

$$= 1 - (n_{oy}^2 \cos^2 \delta - 2n_{oy}n_{oz} \sin \delta \cos \delta + n_{oz}^2 \sin^2 \delta) \quad (B17)$$

$$= 1 - (n_{oy} \cos \delta - n_{oz} \sin \delta)^2 \quad (B18)$$

$$= 1 - n_y^2. \quad (B19)$$

from Eq. (B2). From Eq. (B13)

$$n_z = (1 - n_y^2)^{\frac{1}{2}} \cos(\phi + \beta) \quad (\text{B20})$$

$$\phi = \cos^{-1} \left[\frac{n_z}{(1 - n_y^2)^{\frac{1}{2}}} \right] - \cos^{-1} \left[\frac{(n_{0y} \sin \delta + n_{0z} \cos \delta)}{(1 - n_y^2)^{\frac{1}{2}}} \right]. \quad (\text{B21})$$

Eq. (B21) is a closed form expression for ϕ .

APPENDIX C

LATS PROGRAMS

C1. Setup Routine

```
C      19 NOV 1987
C      PROGRAM TARGETSET.FOR SEPT1, 1987
C      MODIFIED FROM PROGRAM LATS.FOR AUGUST, 1987
C      MUST BE LINKED WITH FILES PFIB, M1SET, M2SET, SCAN
C      PROGRAM SERVES AS AN INTERACTIVE SET UP PROGRAM
C      REQUESTING SENSOR #, MOVING GONIOMETER TO SENSOR LOCUS
C      PROMPTING USER TO MANUALLY POSITION BEAM, SCANNING BEAM
C      ACROSS SENSOR TO MAXIMIZE IRRADIANCE, AND RECORDING
C      THIS POSITION INFORMATION ON DATA FILES
C      THE FOLLOWING IS GPIB PROFESSIONAL FORTRAN HEADER
COMMON /IBGLOB/ IBSTA
integer*2 ibclr, TIMO, END
integer*2 ibfind, IBSTA
integer*2 ibrd, ibrdf, ibtmo
integer*2 ibwrt, ibwrtf
INTEGER*2  TNONE, T10us, T30us, T100us, T300us
INTEGER*2  T1ms, T3ms, T10ms, T30ms, T100ms
INTEGER*2  T300ms, T1s, T3s, T10s, T30s
INTEGER*2  T100s, T300s, T1000s, CMPL
C      VARIABLE DECLARATIONS FOR PROGRAM
INTEGER    XAXIS, YAXIS, GP, DN
INTEGER*2  BD, GPIB0, IOUT, F, ENTRY
INTEGER*2  PPR, SPR, IXX, IYY, IX, IY, PI, DI
REAL       R(3), SH(3), MH(3)
REAL       PH, DL, XP, YP
CHARACTER*2 SE, SS, TS, US, VS, GS
LOGICAL    BRANCH, NEW
c      EOS mode: bit values
      data BIN/Z'1000', XEOS/Z'800', REOS/Z'400'/
c      Timeout values
      data TNONE/0/, T10us/1/, T30us/2/, T100us/3/, T300us/4/
      data T1ms/5/, T3ms/6/, T10ms/7/, T30ms/8/, T100ms/9/
      data T300ms/10/, T1s/11/, T3s/12/, T10s/13/, T30s/14/
      data T100s/15/, T300s/16/, T1000s/17/
      data T100s/15/, LF/10/
C
C      GIVE ONLINE INSTRUCTIONS TO USER
WRITE (*,*) ' TO RUN, THE FOLLOWING FILES MUST BE SUPPLIED:'
WRITE (*,*) ' "HARDWARE.DAT" CONTAINS SYSTEM MEASUREMENTS:'
WRITE (*,*) ' FACTOR (1.0),'
```

```

WRITE (*,*) 'HORIZ DISTANCE TO BEAM ORIGIN FROM PANEL CENTER,'
WRITE (*,*) 'VERT DISTANCE TO BEAM ORIGIN FROM PANEL CENTER,'
WRITE (*,*) 'HORIZ DISTANCE, STEERING TO TRANSFER MIRROR,'
WRITE (*,*) 'VERT DISTANCE, STEERING TO TRANSFER MIRROR,'
WRITE (*,*) 'Z DISTANCE, STEERING TO TRANSFER MIRROR,'
WRITE (*,*) 'ALL DISTANCES IN CM UNITS'
WRITE (*,*) "DISPLAY" CONTAINS THE COMMAND "V 0[CR]"
WRITE (*,*) "RATE.DAT" CONTAINS A,R,S,F GONIOMETER PARAMETERS:
WRITE (*,*) "A[CR]R 250[CR]S 2[CR]F 20[CR]"
OPEN (20, FILE='DECODE.DAT', STATUS='OLD')
CALL SETUP (F,R,SH,MH,XP,YP)
XAXIS= IBFIND ('XAXIS')
YAXIS= IBFIND ('YAXIS')
GP= IBFIND ('GPB2')
C   ASK USER TO SELECT DATA ENTRY MODE
    WRITE (*, *) '
    WRITE (*, *) ' ENTER (1) FOR KEYBOARD, (2) FOR IEEE DATA BUS'
    READ (*, *) ENTRY
C   PERFORM DEVICE CONNECTION CHECK
    WRITE (*,*) 'THE FOLLOWING TWO RESPONSES SHOULD BE "48"'
    WRITE (*,*) 'IF THEY ARE OTHERWISE, POWER DOWN SYSTEM'
    CALL IBRSP (XAXIS, SPR)
    WRITE (*, *) SPR
    IF (SPR.NE.48) GO TO 1000
    CALL IBRSP (YAXIS, SPR)
    WRITE (*, *) SPR
    IF (SPR.NE.48) GO TO 1000
C   LOOP HERE
1   CALL LOC (F,IX,IY,DN,R,SH,MH,BRANCH,XP,YP,SE,ENTRY,GP)
    IF (IX .LT. 0) GO TO 100
    GO TO 110
100  SS='- '//CHAR(13)
    TS='+' //CHAR(13)
    GO TO 150
110  SS='+' //CHAR(13)
    TS='- '//CHAR(13)
150  IF (IY .LT. 0) GO TO 200
    GO TO 210
200  US='- '//CHAR(13)
    VS='+' //CHAR(13)
    GO TO 250
210  US='+' //CHAR(13)
    VS='- '//CHAR(13)
250  GS='G' //CHAR(13)
    IF (BRANCH) GOTO 6
    CALL MOVE2 (IX, IY, XAXIS, YAXIS, SS, TS, US, VS, GS)
    GOTO 7
6    CALL MOVE1 (IX, IY, XAXIS, YAXIS, SS, TS, US, VS, GS)
7    CALL POS (SE,DN,XAXIS,YAXIS,GP,IX,IY,ENTRY)
C   GO TO HOME
    IF (BRANCH) GOTO 8

```



```

      CALL MOVE2 (IX, IY, XAXIS, YAXIS, T$, S$, V$, U$, G$)
      GOTO 9
8     CALL MOVE1 (IX, IY, XAXIS, YAXIS, T$, S$, V$, U$, G$)
9     IF (ENTRY.EQ.3) GOTO 10
      GOTO 1
1000  WRITE (*, 1001)
1001  FORMAT (' DEVICE CHECK FAILURE ')
3     FORMAT (3A)
5     FORMAT (L1)
10    STOP
      END
C     *****
C     *****
C     *****
      SUBROUTINE LOC (F,IX,IY,DN,R,SH,MH,BRANCH,XP,YP,SE,ENTRY,GP)
C     ROUTINE TAKES MEASURED GONIOMETER PARAMETER AND CALCULATES
C     PHI, DELTA FOR GIVEN SENSOR
      REAL CD, SD, NN, RR, D
      REAL R(3), N(3), RH(3), SH(3), NH(3), MH(3)
      REAL X, X1, X2, A, B, DL, PH
      REAL XW, YW, P, XP, YP
      INTEGER*2 DN, I, F, IX, IY, SE
      INTEGER*2 ENTRY, DUM, IAX, IAY
      INTEGER*2 IBRD, IBWRT, IBFIND
      INTEGER    GP
      CHARACTER*2 UV, IR
      CHARACTER*4 MESSAGE
      CHARACTER*6 FNAME, POWER
      CHARACTER*11 LIST
      LOGICAL    BRANCH
      IF (ENTRY.EQ.1) THEN
C     ASK WHICH TYPE OF SENSOR IS SOUGHT
50    WRITE (*, 5)
        READ (*, 6) SE
        WRITE (*, 107)
        READ (*, *) DN
        END IF
        IF (ENTRY.EQ.2) THEN
            CALL IBRD (GP, POWER, 6)
            WRITE (*, *) ' DATA READ FROM BUS 1'
            CALL DECODE (POWER, SE, IAX, IAY, DN)
            WRITE (*, *) SE, DN
            MESSAGE= 'INIT'
            CALL ENCODE (IAX, IAY, SE, MESSAGE, LIST)
            WRITE (*, *) LIST
            CALL IBWRT (GP, LIST, 11)
            END IF
            CLOSE (7)
            IF (SE.LE.3.AND.SE.GT.0) THEN
                FNAME= 'UV.DAT'
            ELSE IF (SE.EQ.4.OR.SE.EQ.5) THEN

```

```

FNAME= 'IR.DAT'
ELSE
GO TO 50
END IF
OPEN (7, FILE= FNAME, STATUS= 'OLD')
C
52 READ (7,*) I, YW, XW
   IF (DN.EQ.I) GO TO 53
   GO TO 52
53 CLOSE (7)
   R(1)= XW / F
   R(1)= R(1) - XP
   R(2)= YW / F
   R(2)= R(2) - YP
   RR= SQRT (R(1)**2 + R(2)**2 + R(3)**2)
   DO 55 I=1,3
   RH(I)= R(I) / RR
55 CONTINUE
C
C COMPUTE REQUIRED GONIOMETER MIRROR NORMAL, EQS 1-3
DO 60 I=1,3
N(I)= (RH(I) + SH(I)) / 2
60 CONTINUE
NN= SQRT(N(1)**2 + N(2)**2 + N(3)**2)
DO 65 I=1,3
NH(I)= N(I) / NN
65 CONTINUE
C
C COMPUTE DELTA, EQN 19
D= SQRT (1 - MH(1)**2)
X= NH(2) / D
X1= ACOS(X)
X= NH(3) / D
X2= ACOS(X)
DL= X2 -X1
SD= SIN (DL)
CD= COS (DL)
DL= DL * 180 / 3.14159
C COMPUTE PHI, EQN 24B
A= MH(1)
B= MH(2)*SD + MH(3)*CD
X1= B*NH(1) - A*NH(3)
X2= A*NH(1) + B*NH(3)
P= (X1 / X2)
PH= ATAN (X1/X2)
C CHANGE SIGN OF PHI, GONIOMETER USES POS VALUES FOR CW
PH= -PH * 180 / 3.14159
C CONVERT TO GONIOMETER COUNTS WITH ROUNDING
WRITE (*, 111) PH, DL
DL= DL * 100.0
PH= PH * 100.0

```

```

      IY = ANINT(DL)
      IX = ANINT(PH)
      IF (IX.GT.-100.AND.IX.LT.100) BRANCH = .TRUE.
      IF (IX.GE.100.OR.IX.LE.-100) BRANCH = .FALSE.
      WRITE (*, 110) IX, IY
2      FORMAT (A)
5      FORMAT (' IS SENSOR AR(1),UVH(2),UVL(3),IRH(4),IRL(5)? '/')
6      FORMAT (I1)
107     FORMAT (' DETECTOR NUMBER: ')
110     FORMAT (' NUMBER OF X STEPS ', I5, ' NUMBER OF Y STEPS ', I5)
111     FORMAT (' AZIMUTH (DEG)', F6.2, ' ELEVATION (DEG)', F6.2)
      RETURN
      END
C      *****
C      *****
      SUBROUTINE SETUP (F,R,SH,MH,XP,YP)
C      ROUTINE GETS SCALING FACTOR, GONIOMETER POSTIONS REQUIRED
C      FOR COMPUTATIONS
      INTEGER*2 F
      INTEGER*2 RL
      REAL DTR, PH, DL, CD, SD, CP
      REAL SP, SS, MM, RR, NN, XP, YP
      REAL S(3), M(3), SH(3), R(3)
      REAL RH(3), N(3), NH(3), MH(3)
      RL = 30
      OPEN (2,FILE='OAR.DAT',ACCESS='DIRECT',RECL=RL,
/      FORM='FORMATTED', STATUS='OLD')
      OPEN (3,FILE='UVH.DAT',ACCESS='DIRECT',RECL=RL,
/      FORM='FORMATTED', STATUS='OLD')
      OPEN (4,FILE='UVL.DAT',ACCESS='DIRECT',RECL=RL,
/      FORM='FORMATTED', STATUS='OLD')
      OPEN (8,FILE='IRH.DAT',ACCESS='DIRECT',RECL=RL,
/      FORM='FORMATTED', STATUS='OLD')
      OPEN (9,FILE='IRL.DAT',ACCESS='DIRECT',RECL=RL,
/      FORM='FORMATTED', STATUS='OLD')
      OPEN (10, FILE= 'HARDWARE.DAT', STATUS= 'OLD')
      READ (10, *) F, XP, YP, S(1), S(2), S(3), PH, DL
C      ITERATIVE MIRROR NORMAL SOLUTION
C      RUNS 500 ITERATIONS
      DTR = 3.14159 / 180

C      INPUT STEERING MIRROR COORDINATES
C      INPUT GONIOMETER ANGLES AT R=(0, 0, -1)
C      CONVERT TO RADIANS, COMPUTE TRIG FUNCTIONS
      DL = DL * DTR
      PH = PH * DTR
      CD = COS (DL)
      SD = SIN (DL)
      CP = COS (PH)
      SP = SIN (PH)

```

```

C      NORMALIZE STEERING MIRROR VECTOR
      SS= SQRT(S(1)**2 + S(2)**2 + S(3)**2)
      DO 10 I=1,3
      S(I)= S(I) / SS
10     CONTINUE
      DO 25 J=1, 500
C      CALCULATE GONIOMETER MIRROR NORMAL
      MM= SQRT (2* (1-S(3)))
      M(1)= (S(1)*CP - (S(3)-1)*SP)/MM
      M(2)= (S(1)*SP*SD + S(2)*CD + (S(3)-1)*CP*SD)/MM
      M(3)= (S(1)*SP*SD - S(2)*SD + (S(3)-1)*CP*CD)/MM

C      COMPUTE NEW S FROM CALCULATED GONIOMETER MIRROR NORMAL
C      AND MEASURED PHI, DELTA
      S(1)= M(1) * CP + M(2) * SP * SD + M(3) * SP * CD
      S(2)=          M(2) * CD      - M(3) * SD
      S(3)=-M(1) * SP + M(2) * CP * SD + M(3) * CP * CD
25     CONTINUE
C
C      COMPUTE REST MIRROR NORMAL N0 FROM STEERING MIRROR
C      AND TARGET VECTORS
      SS=SQRT (S(1)**2 + S(2)**2 + S(3)**2)
      DO 30 I=1,3
      SH(I)= S(I) / SS
30     CONTINUE
C      INPUT AND NORMALIZE TARGET VECTOR
      R(1)= 0.0
      R(2)= 0.0
      R(3)= -1.0
      RR= 1.0
C      READ (*, *) R(1), R(2), R(3)
C      RR= SQRT (R(1)**2 + R(2)**2 + R(3)**2)
      DO 35 I=1,3
      RH(I)= R(I) / RR
35     CONTINUE
C      COMPUTE REQUIRED GONIOMETER MIRROR VECTOR FROM EQS 1-3
      DO 40 I=1,3
      N(I)= (RH(I) + SH(I)) / 2
40     CONTINUE
      NN= SQRT(N(1)**2 + N(2)**2 + N(3)**2)
      DO 45 I=1,3
      NH(I)= N(I) / NN
45     CONTINUE
      DL= 0
      PH= 0
      SD= SIN (DL)
      CD= COS (DL)
      SP= SIN (PH)
      CP= COS (PH)
C
C      APPLY INVERSE OF TRANSFORMATION IN EQS 8,10

```

```

MH(1)= NH(1) * CP - NH(3) * SP
MH(2)= NH(1) * SP * SD + NH(2) * CD + NH(3) * CP * SD
MH(3)= NH(1) * SP * CD - NH(2) * SD + NH(3) * CP * CD

```

```

C
C   COMPUTE GONIOMETER ANGLES TO GIVEN TARGET COORDINATES
M(1)= MH(1)
M(2)= MH(2)
M(3)= MH(3)
MM= SQRT (M(1)**2 + M(2)**2 + M(3)**2)
DO 50 I=1,3
MH(I)= M(I) / MM
50 CONTINUE
C
C   WRITE (*, 108)
READ (10, *) R(3)
11 FORMAT (A)
12 FORMAT (I2)
RETURN
END
C *****
C *****
SUBROUTINE POS (SE,DN,XAXIS,YAXIS,GP,PL,DL,ENTRY)
INTEGER          XAXIS,YAXIS,GP
INTEGER          DN
INTEGER*2        TIMO, RN, SE, ENTRY,IPH,IDL
INTEGER*2        PL, DL, PM, DM, PS, DS, PD, DD
INTEGER*2        IBWRT, IBRD
C   PL, DL ARE PHI, DELTA OF LOCUS
C   PM, DM ARE PHI, DELTA OF MANUAL POSITION
C   PS, DS ARE PHI, DELTA OF SCANNED MAXIMUM
C   PD, DD ARE PHI, DELTA DIFFERENCES
LOGICAL          BRANCH
CHARACTER*2      $$, US, GS
CHARACTER*6      PAUSE
CHARACTER*7      BUFX, BUFY
CHARACTER*9      CONV
LOGICAL          FLAG, FLAF
LOGICAL          OP
CHARACTER*10     DRT, FVAR
CHARACTER*11     LIST
INTEGER          RCL
FLAF=.FALSE.
C   ASK USER TO MANUALLY POSITION GONIOMETER
WRITE (*, 3)
IF (ENTRY.EQ.1) THEN
PAUSE
GOTO 20
END IF
CALL IBRD (GP, PAUSE, 6)
LIST='#'//99'/'/'/'7'/'/'&'/'/'REDI'/'/'@'
CALL IBWRT (GP, LIST, 11)

```

```

C   SCAN ACROSS SENSOR TO IDENTIFY POSITION OF MAX IRRADIANCE
20  CALL SCAN (XAXIS, YAXIS, GP, PD, DD, ENTRY)
C   SCAN RETURNS VALUES PS, DS DISTANCES TO MAXIMUM
    FLAG=.FALSE.
C   NOW MOVE TO MAXIMUM
    GO TO 50
55  CONTINUE
C   IPH=-10
C   IDL=10
C   S$='- '//CHAR(13)
C   U$='+'//CHAR(13)
C   G$='G'//CHAR(13)
C   CALL IBCLR (XAXIS)
C   CALL IBCLR (YAXIS)
C   CALL MOVE1 (IPH, IDL, XAXIS, YAXIS, S$, T$, U$, V$, G$)
    CALL IBCLR (XAXIS)
    CALL IBCLR (YAXIS)
    CALL IBWRTF (XAXIS, 'DISPLAY')
    CALL IBWRTF (YAXIS, 'DISPLAY')
    CALL IBWAIT (XAXIS, TIMO)
    CALL IBWAIT (YAXIS, TIMO)
    CALL IBRD (XAXIS, BUFX, 9)
    CALL IBRD (YAXIS, BUFY, 9)
    RN= DN
    IF (SE.EQ.1) WRITE (2, 5, REC=RN) DN, BUFX, BUFY
    IF (SE.EQ.2) WRITE (3, 5, REC=RN) DN, BUFX, BUFY
    IF (SE.EQ.3) WRITE (4, 5, REC=RN) DN, BUFX, BUFY
    IF (SE.EQ.4) WRITE (8, 5, REC=RN) DN, BUFX, BUFY
    IF (SE.EQ.5) WRITE (9, 5, REC=RN) DN, BUFX, BUFY
C   READ CHAR STRING INTO INTEGER FORMAT
    WRITE (CONV, 7) BUFX
    READ (CONV, 8) PS
    WRITE (CONV, 7) BUFY
    READ (CONV, 8) DS
    WRITE (*, 1003) PS, DS
    FLAG=.TRUE.
    FLAF=.FALSE.
    PD= PL - PS
    DD= DL - DS
50  IF (PD.GT.-100.AND.PD.LT.100) BRANCH= .TRUE.
    IF (PD.GE.100.OR.PD.LE.-100) BRANCH= .FALSE.
    IF (PD .LT. 0) GO TO 100
    GO TO 110
100  S$='- '//CHAR(13)
    GO TO 150
110  S$='+'//CHAR(13)
150  IF (DD .LT. 0) GO TO 200
    GO TO 210
200  U$='- '//CHAR(13)
    GO TO 250
210  U$='+'//CHAR(13)

```

```

250  G$='G'//CHAR(13)
      IF (FLAF) GOTO 51
      IF (FLAG) THEN
        WRITE (*, 6)
        IF (ENTRY.EQ.1) THEN
          PAUSE
          GOTO 21
        END IF
        CALL IBRD (GP, PAUSE, 6)
        LIST= '#/'99'/'': '/'7'/'&'/'REDI'/'@'
        CALL IBWRT (GP, LIST, 11)
21    WRITE (*, 1002) PL, PS, PD
        WRITE (*, 1002) DL, DS, DD
        FLAF=.TRUE.
        GOTO 50
      END IF
51    IF (BRANCH) GO TO 9
        CALL MOVE2 (PD,DD,XAXIS,YAXIS,SS,TS,US,V$,G$)
        IF (FLAG) GOTO 10
        GOTO 55
9     CALL MOVE1 (PD,DD,XAXIS,YAXIS,SS,TS,US,V$,G$)
        IF (FLAG) GOTO 10
        GOTO 55
2     FORMAT (I1)
3     FORMAT ('0MANUALLY POSITION GONIOMETER TO DESCRIBED TARGET '/')
5     FORMAT (I5,2A9)
6     FORMAT (' WAITING TO RETURN TO HOME POSITION '/')
7     FORMAT (A9)
8     FORMAT (I9)
1000  FORMAT (' ',2A9)
1002  FORMAT (' LOCUS: ', I5, ' SENSOR: ', I5, ' DIFFERENCE: ', I5)
1003  FORMAT (' X POS OF MAX: ', I5, ' Y POS OF MAX: ', I5)
10    RETURN
      END
C     *****
C     *****
      SUBROUTINE MOVE1 (IX,IY,XAXIS,YAXIS,SS,TS,US,V$,G$)
C     ROUTINE TAKES RETURNED STEP COUNTS AND DIRECTS CONTROLLER
C     TO MOVE GONIOMETER TO SELECTED TARGET
C     THE FOLLOWING IS GPIB PROFESSIONAL FORTRAN HEADER
      integer*2 ibclr
      integer*2 ibrd
      integer*2 ibwrt, TIMO, END
      integer*2 BIN, XEOS, REOS, LF, S, CMPL
      INTEGER*2 TNONE, T10us, T30us, T100us, T300us
      INTEGER*2 T1ms, T3ms, T10ms, T30ms, T100ms
      INTEGER*2 T300ms, T1s, T3s, T10s, T30s
      INTEGER*2 T100s, T300s, T1000s
      REAL      XSTEPS, YSTEPS, PH, DL
      INTEGER   XAXIS, YAXIS, IOUT
      INTEGER*2 DN, A, L, K, J

```

```

INTEGER*2  IXX, IYY, IX, IY
CHARACTER*1  N$
CHARACTER*2  S$, T$, U$, V$, G$, BAD
CHARACTER*4  X4$, Y4$
CHARACTER*5  X3$, Y3$
CHARACTER*6  X$, Y$, X1$, Y1$
CHARACTER*7  X2$, Y2$
CHARACTER*8  RATE

```

c GPIB Commands: values

```
DATA TIMO/Z'4000'/, END/Z'2000'/
```

c Iberr error messages: values

c EOS mode: bit values

```
data BIN/Z'1000'/, XEOS/Z'800'/, REOS/Z'400'/
```

c Timeout values

```
data TNONE/0/, T10us/1/, T30us/2/, T100us/3/, T300us/4/
```

```
data T1ms/5/, T3ms/6/, T10ms/7/, T30ms/8/, T100ms/9/
```

```
data T300ms/10/, T1s/11/, T3s/12/, T10s/13/, T30s/14/
```

```
data T100s/15/, T300s/16/, T1000s/17/
```

```
data S/08/, LF/10/
```

```
RATE= 'RATE.DAT'
```

```
A= DN
```

1 FORMAT (' OUTPUT DATA FILE NAME: ')

11 FORMAT (A)

16 FORMAT (I4)

17 FORMAT (I3)

18 FORMAT (I2)

19 FORMAT (I1)

C RATE IS THE FILE WHICH CONTAINS A,R,S,F STATEMENTS

```
CALL IBCLR (XAXIS)
```

```
CALL IBWRTF (XAXIS,RATE)
```

```
CALL IBCLR (YAXIS)
```

```
CALL IBWRTF (YAXIS,RATE)
```

C

```
IXX=IABS(IX)
```

```
IYY=IABS(IY)
```

```
N$='N'
```

```
IF (IXX.LT.1000 .AND. IXX .GE. 100) WRITE(X$,17) IXX
```

```
IF (IXX.GE.1000) WRITE(X$,16) IXX
```

```
IF (IYY.LT.1000 .AND. IYY .GE. 100) WRITE(Y$,17) IYY
```

```
IF (IYY.GE.1000) WRITE(Y$,16) IYY
```

```
IF (IXX.LT.100 .AND. IXX .GE. 10) WRITE (X$,18) IXX
```

```
IF (IYY.LT.100 .AND. IYY .GE. 10) WRITE (Y$,18) IYY
```

```
IF (IXX.LT.10) WRITE (X$, 19) IXX
```

```
IF (IYY.LT.10) WRITE (Y$, 19) IYY
```

```
IF (IXX.EQ.0) X$='00'
```

```
IF (IYY.EQ.0) Y$='00'
```

```
X1$=N$/' '//X$(1:3)//CHAR(13)
```

```
X2$=N$/' '//X$(1:4)//CHAR(13)
```

```
X3$=N$/' '//X$(1:2)//CHAR(13)
```

```
X4$=N$/' '//X$(1:1)//CHAR(13)
```



```

Y1$=N$/' '//Y$(1:3)//CHAR(13)
Y2$=N$/' '//Y$(1:4)//CHAR(13)
Y3$=N$/' '//Y$(1:2)//CHAR(13)
Y4$=N$/' '//Y$(1:1)//CHAR(13)
IF(IXX.LT.100.AND.IXX.GE.10.AND.IYY.LT.100.AND.IYY.GE.10) GOTO 800
IF (IXX.LT.100.AND.IXX.GE.10.AND.IYY.GE.1000) GO TO 900
IF(IXX.LT.100.AND.IXX.GE.10.AND.IYY.GE.100.AND.IYY.LT.1000)GOTO 12
IF (IXX.LT.10.AND.IYY.LT.10) GO TO 1300
IF (IXX.LT.10.AND.IYY.GE.10.AND.IYY.LT.100) GO TO 1400
IF (IXX.LT.10.AND.IYY.GE.100.AND.IYY.LT.1000) GO TO 1500
IF (IXX.LT.10.AND.IYY.GE.1000) GO TO 1600
IF (IXX.GE.10.AND.IXX.LT.100.AND.IYY.LT.10) GO TO 1700
800  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X3$,5)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y3$,5)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
900  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X3$,5)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y2$,7)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
12   CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X3$,5)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y1$,6)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
1300 CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X4$,4)
      CALL IBWRT(XAXIS,S$,2)

```

```
CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
CALL IBWAIT (XAXIS, TIMO)
CALL IBCLR(YAXIS)
CALL IBWRT(YAXIS,Y4$,4)
CALL IBWRT(YAXIS,U$,2)
CALL IBWRT(YAXIS,G$,2)
GO TO 300
C   X AXIS MOVEMENT
1400 CALL IBCLR(XAXIS)
CALL IBWRT(XAXIS,X4$,4)
CALL IBWRT(XAXIS,S$,2)
CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
CALL IBWAIT (XAXIS, TIMO)
CALL IBCLR(YAXIS)
CALL IBWRT(YAXIS,Y3$,5)
CALL IBWRT(YAXIS,U$,2)
CALL IBWRT(YAXIS,G$,2)
GO TO 300
C   X AXIS MOVEMENT
1500 CALL IBCLR(XAXIS)
CALL IBWRT(XAXIS,X4$,4)
CALL IBWRT(XAXIS,S$,2)
CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
CALL IBWAIT (XAXIS, TIMO)
CALL IBCLR(YAXIS)
CALL IBWRT(YAXIS,Y1$,6)
CALL IBWRT(YAXIS,U$,2)
CALL IBWRT(YAXIS,G$,2)
GO TO 300
C   X AXIS MOVEMENT
1600 CALL IBCLR(XAXIS)
CALL IBWRT(XAXIS,X4$,4)
CALL IBWRT(XAXIS,S$,2)
CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
CALL IBWAIT (XAXIS, TIMO)
CALL IBCLR(YAXIS)
CALL IBWRT(YAXIS,Y2$,7)
CALL IBWRT(YAXIS,U$,2)
CALL IBWRT(YAXIS,G$,2)
GO TO 300
C   X AXIS MOVEMENT
1700 CALL IBCLR(XAXIS)
CALL IBWRT(XAXIS,X3$,5)
CALL IBWRT(XAXIS,S$,2)
CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
CALL IBWAIT (XAXIS, TIMO)
```

```

CALL IBCLR(YAXIS)
CALL IBWRT(YAXIS,Y4$,4)
CALL IBWRT(YAXIS,U$,2)
CALL IBWRT(YAXIS,G$,2)
GO TO 300

```

```

C   X AXIS MOVEMENT
300 CONTINUE
    RETURN
    END

```

```

C   *****
C   *****

```

```

C   SUBROUTINE MOVE2 (IX,IY,XAXIS,YAXIS,SS,TS,U$,V$,G$)
C   ROUTINE TAKES RETURNED STEP COUNTS AND DIRECTS CONTROLLER
C   TO MOVE GONIOMETER TO SELECTED TARGET
C   THE FOLLOWING IS GPIB PROFESSIONAL FORTRAN HEADER

```

```

integer*2 ibclr
integer*2 ibrd
integer*2 ibwrt
integer*2 BIN, XEOS, REOS, TIMO, END, S, LF
INTEGER*2  TNONE, T10us,T30us,T100us,T300us
INTEGER*2  T1ms,T3ms,T10ms,T30ms,T100ms
INTEGER*2  T300ms,T1s,T3s,T10s,T30s
INTEGER*2  T100s,T300s,T1000s
REAL      XSTEPS, YSTEPS, PH, DL
INTEGER    XAXIS, YAXIS, IOUT
INTEGER*2  DN, A, L, K, J
INTEGER*2  IXX, IYY, IX, IY
CHARACTER*1  NS
CHARACTER*2  SS, TS, U$, V$, G$, BAD
CHARACTER*4  X4$, Y4$
CHARACTER*5  X3$, Y3$
CHARACTER*6  X$, Y$, X1$, Y1$
CHARACTER*7  X2$, Y2$
CHARACTER*8  RATE

```

c GPIB Commands: values

```
DATA TIMO/Z'4000'/, END/Z'2000'/
```

c Ibrerr error messages: values

c EOS mode: bit values

```
data BIN/Z'1000'/,XEOS/Z'800'/,REOS/Z'400'/
```

c Timeout values

```
data TNONE/0/,T10us/1/,T30us/2/,T100us/3/,T300us/4/
```

```
data T1ms/5/,T3ms/6/,T10ms/7/,T30ms/8/,T100ms/9/
```

```
data T300ms/10/,T1s/11/,T3s/12/,T10s/13/T30s/14/
```

```
data T100s/15/,T300s/16/,T1000s/17/
```

```
data S/08/, LF/10/
```

```
RATE= 'RATE.DAT'
```

```
A= DN
```

```
11  FORMAT (A)
```

```
16  FORMAT (I4)
```

```
17  FORMAT (I3)
```

```

18  FORMAT (I2)
19  FORMAT (I1)
C   RATE IS THE FILE WHICH CONTAINS A,R,S,F STATEMENTS
    CALL IBCLR (XAXIS)
    CALL IBWRTF (XAXIS,RATE)
    CALL IBCLR (YAXIS)
    CALL IBWRTF (YAXIS,RATE)

C
    IXX=IABS(IX)
    IYY=IABS(IY)
    NS='N'
    IF (IXX.LT.1000 .AND. IXX .GE. 100) WRITE(X$,17) IXX
    IF (IXX.GE.1000) WRITE(X$,16) IXX
    IF (IYY.LT.1000 .AND. IYY .GE. 100) WRITE(Y$,17) IYY
    IF (IYY.GE.1000) WRITE(Y$,16) IYY
    IF (IXX.LT.100.AND.IXX.GE.10) WRITE (X$,18) IXX
    IF (IYY.LT.100.AND.IXX.GE.10) WRITE (Y$,18) IYY
    IF (IXX.LT.10) WRITE (X$, 19) IXX
    IF (IYY.LT.10) WRITE (Y$, 19) IYY
    IF (IXX.EQ.0) X$='00'
    IF (IYY.EQ.0) Y$='00'
    X1$=NS// '/'//X$(1:3)//CHAR(13)
    X2$=NS// '/'//X$(1:4)//CHAR(13)
    X3$=NS// '/'//X$(1:2)//CHAR(13)
    X4$=NS// '/'//X$(1:1)//CHAR(13)
    Y1$=NS// '/'//Y$(1:3)//CHAR(13)
    Y2$=NS// '/'//Y$(1:4)//CHAR(13)
    Y3$=NS// '/'//Y$(1:2)//CHAR(13)
    Y4$=NS// '/'//Y$(1:1)//CHAR(13)
    IF (IXX.LT.1000.AND.IXX.GE.100.AND.IYY.GE.1000) GO TO 500
    IF (IXX.GE.1000.AND.IYY.LT.1000.AND.IYY.GE.100) GO TO 600
    IF (IXX .GE. 1000 .AND. IYY .GE. 1000) GO TO 700
    IF(IXX.GE.100.AND.IXX.LT.1000.AND.IYY.LT.100.AND.IYY.GE.10)GOTO 10
    IF (IXX.GE.1000.AND.IYY.LT.100.AND.IYY.GE.10) GO TO 1100
    IF (IXX.GE.100.AND.IXX.LT.1000.AND.IYY.LT.10) GO TO 1800
    IF (IXX.GE.1000.AND.IYY.LT.10) GO TO 1900
    IF(IXX.LT.1000.AND.IXX.GE.100.AND.IYY.LT.1000.AND.IYY.GE.100)
      /  GOTO 20
C   X AXIS MOVEMENT
500  CALL IBCLR(XAXIS)
    CALL IBWRT(XAXIS,X1$,6)
    CALL IBWRT(XAXIS,S$,2)
    CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
    CALL IBWAIT (XAXIS, TIMO)
    CALL IBCLR(YAXIS)
    CALL IBWRT(YAXIS,Y2$,7)
    CALL IBWRT(YAXIS,U$,2)
    CALL IBWRT(YAXIS,G$,2)
    GO TO 300
C   X AXIS MOVEMENT

```

```

600  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y1$,6)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
2    GO TO 300
C    X AXIS MOVEMENT
700  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y2$,7)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
10   CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X1$,6)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y3$,5)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
1100 CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y3$,5)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
1800 CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X1$,6)
      CALL IBWRT(XAXIS,S$,2)

```

```

      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y4$,4)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
1900  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y4$,4)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
20   CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X1$,6)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y1$,6)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
300  CONTINUE
      RETURN
      END
C    .....
C    .....
SUBROUTINE SCAN (XAXIS, YAXIS, GP, IX, IY, ENTRY)
C    ROUTINE STEPS BEAM THROUGH SCANNING OF SENSOR AND DETERMINES
C    POSITION OF MAXIMUM IRRADIANCE
      INTEGER          XAXIS, YAXIS, GP
      INTEGER*2  XAR, YAR, MXAR, MYAR
      INTEGER*2  IPH, IDL, IX, IY, DUM
      INTEGER*2  IBTMO, IBCLR, IBRD, TIMO, IBWRT
      INTEGER*2  T3MS, T3S, SPR
      INTEGER*2  ITYPE, ENTRY
      REAL        ARRAY, POWVAL
      CHARACTER*2  SS, US, GS
      CHARACTER*4  MESSAGE
      CHARACTER*6  POWER
      CHARACTER*11 LIST
      DIMENSION ARRAY (1:5, 1:5)

```

```

DATA      T3MS/6/,T1S/11/,T3S/12/
C  PERFORM DEVICE CHECK
  CALL IBRSP (GP, SPR)
  WRITE (*, *) SPR
  IF (SPR.NE.0) GO TO 90
C  MOVE BEAM FROM MANUALLY SELECTED POSITION TO BEGINNING OF SCAN
  IPH=0
  IDL=0
  SS='- '//CHAR(13)
  US='+ '//CHAR(13)
  GS='G'//CHAR(13)
  CALL IBCLR (XAXIS)
  CALL IBCLR (YAXIS)
  CALL MOVE1 (IPH, IDL, XAXIS, YAXIS, SS, TS, US, VS, GS)
  IF (ENTRY.EQ.1) GOTO 31
  CALL IBRD (GP, POWER, 6)
  WRITE (*, 70) POWER
  CALL DECODE (POWER, ITYPE, XAR, YAR, DN)
  MESSAGE= 'REDI'
  IF ((XAR.NE.1).OR.(YAR.NE.1)) MESSAGE='BADD'
  WRITE (*, *) 'XAR AND YAR ARE:'
  WRITE (*, *) XAR,YAR
  CALL ENCODE (XAR, YAR, ITYPE, MESSAGE, LIST)
  CALL IBWRT (GP, LIST, 11)
31  US='- '//CHAR(13)
C  SCAN THROUGH 25 STEPS IN 5 INCREMENTS OF 5 STEPS EACH
  DO 10 J=5
  DO 20 I=1
  IF (ENTRY.EQ.1) GOTO 32
  CALL IBRD (GP, POWER, 6)
  CALL DECODE (POWER, ITYPE, XAR, YAR, DN)
  MESSAGE= 'REDI'
32  K= J/2
  IF (MOD(I,2).NE.0) THEN
    SS='+ '//CHAR(13)
    M=I+1
  END IF
  IF (MOD(I,2).EQ.0) THEN
    SS='- '//CHAR(13)
    M=I-1
  END IF
  N=J/5
  IPH=0
  IDL=0
  IF (ENTRY.EQ.1) GOTO 33
  IF (XAR.NE.M.OR.YAR.NE.N) MESSAGE= 'BADD'
33  CALL MOVE1 (IPH, IDL, XAXIS, YAXIS, SS, TS, US, VS, GS)
  IF (ENTRY.EQ.1) GOTO 20
  CALL ENCODE (XAR, YAR, ITYPE, MESSAGE, LIST)
  CALL IBWRT (GP, LIST, 11)
20  CONTINUE

```

```

IF (J.EQ.5) GOTO 11
IPH=0
IDL=0
IF (ENTRY.EQ.1) GOTO 34
CALL IBRD (GP, POWER, 6)
CALL DECODE (POWER, ITYPE, XAR, YAR, DN)
MESSAGE= 'REDI'
34 N= N+1
IF (ENTRY.EQ.1) GOTO 35
IF (XAR.NE.M.OR.YAR.NE.N) MESSAGE= 'BADD'
35 CALL MOVE1 (IPH, IDL, XAXIS, YAXIS, $$, T$, U$, V$, G$)
IF (ENTRY.EQ.1) GOTO 10
CALL ENCODE (XAR, YAR, ITYPE, MESSAGE, LIST)
CALL IBWRT (GP, LIST, 11)
10 CONTINUE
11 IF (ENTRY.EQ.1) THEN
MXAR= 1
MYAR= 1
GOTO 36
END IF
CALL IBRD (GP, POWER, 6)
CALL DECODE (POWER, ITYPE, MXAR, MYAR, DN)
IF (XAR.LT.1.OR.XAR.GT.5.OR.YAR.LT.1.OR.YAR.GT.5) MESSAGE= 'BADD'
IF (ITYPE.GT.9.OR.ITYPE.LT.8) MESSAGE= 'BADD'
IF (ITYPE.EQ.9) ENTRY=3
CALL ENCODE (MXAR, MYAR, ITYPE, MESSAGE, LIST)
CALL IBWRT (GP, LIST, 11)
36 IX= -(5*(1-MXAR))
IY= (5*(1-MYAR))
RETURN
70 FORMAT (A10)
80 FORMAT (I10)
90 WRITE (*,*) ' DEVICE CHECK ERROR'
END
C .....
C .....
SUBROUTINE DECODE (POWER, ITYPE, IDX, INY, DN)
INTEGER*2 ITYPE, IDX, INY, DN
CHARACTER*1 TYPE, DX, NY
CHARACTER*6 POWER, CONV
WRITE (20, 1) POWER
TYPE= POWER (2:2)
DX= POWER (4:4)
NY= POWER (5:5)
WRITE (CONV, 2) TYPE
READ (CONV, 3) ITYPE
WRITE (*, *) ITYPE
WRITE (CONV, 2) DX
READ (CONV, 3) IDX
WRITE (*, *) IDX
WRITE (CONV, 2) NY

```



```

      READ (CONV,3) INY
      WRITE (*, *) INY
      DN=IDX*10
      DN=DN+INY
      WRITE (*, *) DN
1     FORMAT (A6)
2     FORMAT (A)
3     FORMAT (I1)
      RETURN
      END
C     *****
      SUBROUTINE ENCODE (IPHI, IDEL, ITYPE, MESSAGE, LIST)
      INTEGER*2  IPHI, IDEL, ITYPE
      CHARACTER*1 PHI, DEL, TYPE, CONV
      CHARACTER*4 MESSAGE
      CHARACTER*11 LIST
      WRITE (CONV, 1) IPHI
      READ (CONV,2) PHI
      WRITE (CONV, 1) IDEL
      READ (CONV, 2) DEL
      WRITE (CONV, 1) ITYPE
      READ (CONV, 2) TYPE
      LIST= '#//PHI//DEL//'://TYPE//& '//MESSAGE//'@'
1     WRITE (*, 2) LIST
2     FORMAT (I1)
      FORMAT (A)
      RETURN
      END

```

C2. Pointing Routine

```

C   FROM PROGRAM HITRGT.FOR SEPT 15, 1987
C   LINK WITH PFIB;
C   PROGRAM TO READ TARGET POSITION DATA FILES:
C   OAR.DAT, UVH.DAT, UVL.DAT, IRH.DAT, IRL.DAT,
C   FROM DATA RECEIVED FROM LACE MVAX, AND TO THEN
C   EXECUTE POSITIONING OF LASER BEAM ON GIVEN TARGET
C   PROGRAM ALSO TAKES POWER MEASUREMENT, CALIBRATES AND
C   RETURNS POWER TO LACE MVAX.
C   GPIB PROFESSIONAL FORTRAN HEADER
C   COMMON /IBGLOB/ IBSTA, IBERR, IBCNT
C   GPIB FUNCTIONS
C   INTEGER*2  IBSTA, IBERR, IBCNT
C   INTEGER*2  IBCLR, IBCMD, IBEOS, IBFIND, IBONL
C   INTEGER*2  IBRD, IBRDF, IBRSP, IBSRE, IBTMO
C   INTEGER*2  IBWAIT, IBWRT, IBWRTF
C   GPIB COMMANDS
C   INTEGER*2  UNL, UNT, GTL, SDC, PPC, GET, TCT, SPR
C   INTEGER*2  LLO, DCL, PPU, SPE, SPD, PPE, PPD
C   INTEGER*2  ERR, TIMO, END, SRQI, RQS, CMPL, LOK
C   INTEGER*2  REM, CIC, ATN, TACS, LACS, DTAS, DCAS
C   TIMEOUT VALUES
C   INTEGER*2  TNONE, T10US, T30US, T100US, T300US
C   INTEGER*2  T1MS, T3MS, T10MS, T30MS, T100MS
C   INTEGER*2  T300MS, T1S, T3S, T10S, T30S
C   INTEGER*2  T100S, T300S, T1000S
C   MAIN PROGRAM VARIABLE DECLARATIONS
C   INTEGER      XAXIS, YAXIS, ADC, GP
C   INTEGER*2    PH, DL, IX, IY, DN, DATA
C   INTEGER*2    ITYPE, VAL, LASER, LS
C   REAL         POUT, BEAM, PRF, PL, RATIO
C   CHARACTER*6   BUF
C   CHARACTER*1   TYPE, CONV
C   CHARACTER*2   CDN, DONV
C   CHARACTER*7   BONV, CPOUT
C   CHARACTER*10  FNAME
C   CHARACTER*14  LIST
C   LOGICAL      OP, EX
C   GPIB COMMANDS: VALUES
C   DATA         UNL/03/, UNT/95/, GTL/01/, SDC/04/, PPC/05/
C   DATA GET/08/, TCT/09/, LLO/17/, DCL/20/, PPU/21/
C   DATA SPE/24/, SPD/25/, PPE/96/, PPD/112/
C   GPIB STATUS BIT VECTOR: VALUES
C   DATA TIMO/Z'4000'/, END/Z'2000'/, SRQI/Z'1000'/
C   DATA CIC/Z'20'/, RQS/Z'800'/, CMPL/Z'100'/, LOK/Z'80'/
C   DATA REM/Z'40'/, ATN/Z'10'/, TACS/Z'8'/, LACS/Z'4'/
C   DATA DTAS/Z'2'/, DCAS/Z'1'/
C   EOS MODE: BIT VALUES
C   DATA BIN/Z'1000'/, XEOS/Z'800'/, REOS/Z'400'/
C   TIMEOUT VALUES

```

```

DATA TNONE/0/,T10us/1/,T30us/2/,T100us/3/,T300us/4/
DATA T1ms/5/,T3ms/6/,T10ms/7/,T30ms/8/,T100ms/9/
DATA T300ms/10/,T1s/11/,T3s/12/T10s/13/,T30s/14/
DATA T100s/15/,T300s/16/,T1000s/17/
C  DEVICE ADDRESS LOCATION
    XAXIS= IBFIND ('XAXIS')
    YAXIS= IBFIND ('YAXIS')
    GP= IBFIND ('GPIB2')
    ADC= IBFIND ('ADC')
C  PERFORM DEVICE CONNECTION CHECKS
    CALL IBRSP (XAXIS, SPR)
    WRITE (*, *) SPR
    IF (SPR.NE.48) GOTO 12
    CALL IBRSP (YAXIS, SPR)
    WRITE (*, *) SPR
    IF (SPR.NE.48) GOTO 12
    CALL IBRSP (GP, SPR)
    WRITE (*, *) SPR
    CALL IBRSP (ADC, SPR)
    WRITE (*, *) SPR
C  AN IBWRT WILL HANG ON THE BUS UNTIL MVAX READ ADDRESSES
C  IBMPC TO WRITE TO BUS
C  AN IBRD WILL HANG ON THE BUS UNTIL MVAX WRITES DATA ON BUS
C  ASK USER WHETHER TO EXPECT DATA FROM BUS OR KEYBOARD
    WRITE (*, *) 'EXPECT DATA FROM KEYBOARD (1), OR IEEE BUS (2)'
    READ (*, *) DATA
1  CONTINUE
    CALL SEARCH (PH, DL, DN, ITYPE, DATA, GP)
C  EXECUTE MOVEMENT
    CALL MOVE (PH, DL, XAXIS, YAXIS)
C  TAKE POWER MEASUREMENT READING FROM ADC
    CALL POWER (ADC, ITYPE, POUT)
    IF(DATA.EQ.1) GOTO 1
C  ENCODE POUT AS DATA WORD TO SEND BACK TO MVAX
    WRITE (DONV, 20) DN
    READ (DONV, 21) CDN
    WRITE (CONV, 22) ITYPE
    READ (CONV, 23) TYPE
    WRITE (BONV, 24) POUT
    READ (BONV, 25) CPOUT
    LIST= '#'//CDN//':'//TYPE//&'//CPOUT//'@'
C  SEND POWER MEASUREMENT TO MVAX
    WRITE (*, 26) LIST
    CALL IBWRT (GP, LIST, 14)
C  LOOP BACK, GET ANOTHER SENSOR ID NUMBER
    GO TO 1
20  FORMAT (I2.2)
21  FORMAT (A2)
22  FORMAT (I1)
23  FORMAT (A1)
24  FORMAT (F7.3)

```

```

25  FORMAT (A7)
26  FORMAT ( A)
11  FORMAT (A5)
12  WRITE (*, *) 'DEVICE CHECK FAILURE, POWER DOWN CONTROLLER'
    END
C   *****
    SUBROUTINE SEARCH (PH, DL, DN, ITYPE, K, GP)
C   ROUTINE DECODES DATA READ FROM MVAX, OPENS DATA FILE,
C   RETURNS WITH DN, PH, DL VALUES
    INTEGER*2  PH, DL, DN, DM
    INTEGER*2  ITYPE, K, IBRD, IBWRT
    INTEGER      RL, CODE, GP
    CHARACTER*7  FILNM
    CHARACTER*6  BUF
    CHARACTER*4  DONV
    CHARACTER*1  CONV
    CHARACTER*1  BDN, BDM
    CHARACTER*1  BTYPE
    RL = 30
103  IF (K.EQ.1) THEN
    WRITE (*, 99)
    WRITE (*, 101)
    READ (*, *) ITYPE, DN
    GO TO 102
    END IF
    IF (K.EQ.2) THEN
    CALL IBRD (GP, BUF, 6)
    BTYPE = BUF(2:2)
    BDN = BUF(4:4)
    BDM = BUF(5:5)
    WRITE (CONV, 4) BTYPE
    READ (CONV, 6) ITYPE
    WRITE (CONV, 4) BDN
    READ (CONV, 6) DN
    WRITE (CONV, 4) BDM
    READ (CONV, 6) DM
    DN = DN*10
    DN = DN + DM
    END IF
102  CODE = 85
    IF (ITYPE.EQ.4.OR.ITYPE.EQ.5) CODE = 40
    IF (ITYPE.EQ.1) FILNM = 'OAR.DAT'
    IF (ITYPE.EQ.2) FILNM = 'UVH.DAT'
    IF (ITYPE.EQ.3) FILNM = 'UVL.DAT'
    IF (ITYPE.EQ.4) FILNM = 'IRH.DAT'
    IF (ITYPE.EQ.5) FILNM = 'IRL.DAT'
    IF (ITYPE.LT.1.OR.ITYPE.GT.5) GO TO 3
    WRITE (*, *) FILNM
    OPEN (2, FILE = FILNM, ACCESS = 'DIRECT', RECL = RL,
/   FORM = 'FORMATTED', STATUS = 'OLD')
C   SEARCH FOR PH, DL

```

```

READ (2, REC=DN, FMT=8) J, PH, DL
WRITE (*, *) J, PH, DL
10 CONTINUE
1 RETURN
3 WRITE (*, 9)
  GOTO 103
4 FORMAT (A)
5 FORMAT (A3)
6 FORMAT (I1)
7 FORMAT (A4)
8 FORMAT (I5, 2I9)
9 FORMAT (' SENSOR TYPE NOT PROPERLY IDENTIFIED ')
99 FORMAT (' SENSOR TYPES: (1)AR, (2)UVH, (3)UVL, (4)IRH, (5)IRL')
100 FORMAT (' ENTER (1) FOR USER INPUT,
/          (2) FOR IEEE DATA XFER: ')
101 FORMAT(' ENTER SENSOR TYPE, ID#: ')
END
C *****
SUBROUTINE MOVE (PH, DL, XAXIS, YAXIS)
C ROUTINE TAKES RETURNED STEP COUNTS AND DIRECTS CONTROLLER
C TO MOVE GONIOMETER TO SELECTED TARGET
  INTEGER*2 PH, DL, IXX, IYY, IX, IY
  INTEGER*2 XAXIS, YAXIS, TIMO
  CHARACTER*1 NS
  CHARACTER*2 SS, US, GS
  CHARACTER*4 X4$, Y4$
  CHARACTER*5 X3$, Y3$
  CHARACTER*6 X$, Y$, X1$, Y1$
  CHARACTER*7 X2$, Y2$, BUFX, BUFY
  CHARACTER*8 RATE
  CHARACTER*9 CV
C READ CURRENT POSITION FROM DISPLAY
  CALL IBCLR (XAXIS)
  CALL IBCLR (YAXIS)
  CALL IBWRTF (XAXIS, 'DISPLAY')
  CALL IBWRTF (YAXIS, 'DISPLAY')
  CALL IBWAIT (XAXIS, TIMO)
  CALL IBWAIT (YAXIS, TIMO)
  CALL IBRD (XAXIS, BUFX, 9)
  CALL IBRD (YAXIS, BUFY, 9)
  WRITE (*, 2000) BUFX, BUFY
C READ CHARACTER STRING INTO INTEGER FORMAT
  WRITE (CV, 70) BUFX
  READ (CV, 80) IX
  WRITE (CV, 70) BUFY
  READ (CV, 80) IY
C SUBTRACT PRESENT VALUES FROM DESTINATION POSITION
  PH = PH - IX
  DL = DL - IY
  RATE = 'RATE.DAT'
C RATE IS THE FILE WHICH CONTAINS A,R,S,F STATEMENTS

```

```

CALL IBCLR (XAXIS)
CALL IBWRTF (XAXIS,RATE)
CALL IBCLR (YAXIS)
CALL IBWRTF (YAXIS,RATE)
C  WRITE STEP COUNTS AS STRINGS
    IXX=IABS(PH)
    IYY=IABS(DL)
    N$='N'
    IF (IXX.LT.1000 .AND. IXX .GE. 100) WRITE(X$,17) IXX
    IF (IXX.GE.1000) WRITE(X$,16) IXX
    IF (IYY.LT.1000 .AND. IYY .GE. 100) WRITE(Y$,17) IYY
    IF (IYY.GE.1000) WRITE(Y$,16) IYY
    IF (IXX.LT.100.AND.IXX.GE.10) WRITE (X$,18) IXX
    IF (IYY.LT.100.AND.IYY.GE.10) WRITE (Y$,18) IYY
    IF (IXX.LT.10) WRITE (X$, 19) IXX
    IF (IYY.LT.10) WRITE (Y$, 19) IYY
    IF (IXX.EQ.0) X$='00'
    IF (IYY.EQ.0) Y$='00'
    X1$=N$/' '//X$(1:3)//CHAR(13)
    X2$=N$/' '//X$(1:4)//CHAR(13)
    X3$=N$/' '//X$(1:2)//CHAR(13)
    X4$=N$/' '//X$(1:1)//CHAR(13)
    Y1$=N$/' '//Y$(1:3)//CHAR(13)
    Y2$=N$/' '//Y$(1:4)//CHAR(13)
    Y3$=N$/' '//Y$(1:2)//CHAR(13)
    Y4$=N$/' '//Y$(1:1)//CHAR(13)
    IF (PH .LT. 0) GO TO 100
    GO TO 110
100  S$='- '//CHAR(13)
    GO TO 150
110  S$='+'//CHAR(13)
150  IF (DL .LT. 0) GO TO 200
    GO TO 210
200  U$='- '//CHAR(13)
    GO TO 250
210  U$='+'//CHAR(13)
250  G$='G'//CHAR(13)
    IF (IXX.LT.100.AND.IXX.GE.10.AND.IYY.LT.100.AND.IYY.GE.10)GOTO 800
    IF (IXX.LT.100.AND.IXX.GE.10.AND.IYY.GE.1000) GO TO 900
    IF (IXX.LT.100.AND.IXX.GE.10.AND.IYY.GE.100.AND.IYY.LT.1000)GOTO 12
    IF (IXX.LT.10.AND.IYY.LT.10) GO TO 1300
    IF (IXX.LT.10.AND.IYY.GE.10.AND.IYY.LT.100) GO TO 1400
    IF (IXX.LT.10.AND.IYY.GE.100.AND.IYY.LT.1000) GO TO 1500
    IF (IXX.LT.10.AND.IYY.GE.1000) GO TO 1600
    IF (IXX.GE.10.AND.IXX.LT.100.AND.IYY.LT.10) GO TO 1700
    IF (IXX.LT.1000.AND.IXX.GE.100.AND.IYY.GE.1000) GO TO 500
    IF (IXX.GE.1000.AND.IYY.LT.1000.AND.IYY.GE.100) GO TO 600
    IF (IXX .GE. 1000 .AND. IYY .GE. 1000) GO TO 700
    IF (IXX.GE.100.AND.IXX.LT.1000.AND.IYY.LT.100.AND.IYY.GE.10)GOTO 10
    IF (IXX.GE.1000.AND.IYY.LT.100.AND.IYY.GE.10) GO TO 1100
    IF (IXX.GE.100.AND.IXX.LT.1000.AND.IYY.LT.10) GO TO 1800

```

```

      IF (IXX.GE.1000.AND.IYY.LT.10) GO TO 1900
      IF(IXX.LT.1000.AND.IXX.GE.100.AND.IYY.LT.1000.AND.IYY.GE.100)
/      GOTO 20
800    CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X3$,5)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y3$,5)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT
900    CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X3$,5)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y2$,7)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT
12     CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X3$,5)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y1$,6)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT
1300   CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X4$,4)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y4$,4)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT

```

```

1400  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X4$,4)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y3$,5)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT
1500  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X4$,4)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y1$,6)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT
1600  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X4$,4)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y2$,7)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT
1700  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X3$,5)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C      Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y4$,4)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C      X AXIS MOVEMENT
500   CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X1$,6)
      CALL IBWRT(XAXIS,S$,2)

```



```

      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y2$,7)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
600  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y1$,6)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
700  CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y2$,7)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
10   CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X1$,6)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y3$,5)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C    X AXIS MOVEMENT
1100 CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C    Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)

```

```

CALL IBCLR(YAXIS)
CALL IBWRT(YAXIS,Y3$,5)
CALL IBWRT(YAXIS,U$,2)
CALL IBWRT(YAXIS,G$,2)
GO TO 300
C   X AXIS MOVEMENT
1800 CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X1$,6)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y4$,4)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C   X AXIS MOVEMENT
1900 CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X2$,7)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y4$,4)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
      GO TO 300
C   X AXIS MOVEMENT
20   CALL IBCLR(XAXIS)
      CALL IBWRT(XAXIS,X1$,6)
      CALL IBWRT(XAXIS,S$,2)
      CALL IBWRT(XAXIS,G$,2)
C   Y AXIS MOVEMENT
      CALL IBWAIT (XAXIS, TIMO)
      CALL IBCLR(YAXIS)
      CALL IBWRT(YAXIS,Y1$,6)
      CALL IBWRT(YAXIS,U$,2)
      CALL IBWRT(YAXIS,G$,2)
300  CONTINUE
11   FORMAT (A)
16   FORMAT (I4)
17   FORMAT (I3)
18   FORMAT (I2)
19   FORMAT (I1)
70   FORMAT (A9)
80   FORMAT (I9)
2000 FORMAT (2A9)
      RETURN
      END

```

```

C *****
SUBROUTINE POWER (ADC, ITYPE, POUT)
INTEGER*2  PIN, ADC, ITYPE, LASER
INTEGER          NUM
REAL          RATIO, SCALE, ND
REAL          BEAM, PRF, PL, SENSAR, AREA, POUT
CHARACTER*2    BAD
CHARACTER*10   FILNM
LOGICAL        EX, OP
CALL IBCLR (ADC)
CALL IBWRT (ADC, 'H8AJ', 4)
CALL IBRD (ADC, BAD, 2)
CALL CONVERT (BAD, PIN)
WRITE (*, *) PIN
C WRITE (*, 5)
C READ (*, 6) SCALE
SCALE = 1.0
C WRITE (*, 7)
C READ (*, 8) ND
ND = 1
23 CONTINUE
C WRITE (*, 10)
C READ (*, *) LASER
LASER = 1
OPEN (9, FILE='LASERS.DAT', STATUS='OLD')
DO 21 I=1, 5
READ (9, *) NUM, BEAM, PRF, PL, RATIO
IF (I.EQ.5.AND.NUM.NE.LASER) GO TO 23
IF (NUM.EQ.LASER) GO TO 22
21 CONTINUE
C WHAT IS SENSOR AREA?
22 CLOSE (9)
IF (ITYPE.EQ.1) SENSAR= 1.0
IF (ITYPE.EQ.2) SENSAR= 1.0
IF (ITYPE.EQ.3) SENSAR= 1.0
IF (ITYPE.EQ.4) SENSAR= 1.0
IF (ITYPE.EQ.5) SENSAR= 1.0
C COMPUTE FACTORS
PIN= PIN + 1024
SCALE= SCALE / 1024
ND= 10 ** (-ND)
PRF= 1 / PRF
PL= 1 / PL
AREA= (SENSAR / BEAM)
POUT= PIN * SCALE * ND * PRF * PL * RATIO * AREA
WRITE (*, *) POUT
C FORMAT (' POWER METER FULL SCALE VALUE: ')
C FORMAT (F5.2)
C FORMAT (' NEUTRAL DENSITY VALUE: ')
C FORMAT (F5.2)
C FORMAT (' SELECT LASER: (1)AR, (2)HN, (3)DF, (4)EX, (5)YAG ')

```

```

RETURN
END
C *****
SUBROUTINE CONVERT (BAD,IOUT)
C THIS ROUTINE CONVERTS THE ADC READING TO AN INTEGER
C ROUTINE IS CALLED UP BY SUBROUTINE POWER
  CHARACTER*2 BAD
  CHARACTER*1 BADA,BADB
  INTEGER*2 IA,IB,IC,ID,IE,IGF,IOUT
  BADA=BAD(:1)
  BADB=BAD(2:)
  IA=ICHAR(BADA)
  IB=ICHAR(BADB)
  IF (IA .LT. 4) GO TO 200
100  IC = 255 - IA
      ID = IC*256
      IE = 256 - IB
      IGF = ID + IE
      IOUT = -1 * IGF
      GO TO 300
200  IC = IA * 256
      IOUT = IC + IB
300  RETURN
      END

```